НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ І
СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ

«На правах рукопису»
УДК 004.4'4

«До захисту допущено»

Завідувач кафедри СПСКС

_____ В.П.Тарасенко_
(підпис) (ініціали, прізвище)

"___"_____2018р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 123 Комп'ютерна інженерія
Спеціалізовані комп'ютерні системи

на тему: Спосіб структуризації програмно-конфігуровних мереж SDN

Виконав: студент ІІ курсу, групи КВ-63м

Єрмоленко Ігор Андрійович _____ _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник доцент, к.т.н, Щербина О.А. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – другий (магістерський)
Спеціальність 123 Комп'ютерна інженерія
Спеціалізовані комп'ютерні системи

ЗАТВЕРДЖУЮ
Завідувач кафедри СПСКС
_____ _В.П.Тарасенко__
(підпис) (ініціали, прізвище)

«___»_____2018р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Єрмоленко Ігор Андрійович
(прізвище, ім'я, по батькові)

1. Тема дисертації: СПОСІБ СТРУКТУРИЗАЦІЇ ПРОГРАМНО-КОНФІГУРОВНИХ МЕРЕЖ SDN,
науковий керівник дисертації Щербина Олександр Андрійович, к.т.н., доцент, затверджені наказом по університету від «22» березня 2018 р. №986-с

2. Термін подання студентом дисертації 11 травня 2018 р. _____

3. Об'єкт дослідження: програмно-конфігуровані мережі.

4. Предмет дослідження: спосіб побудови рівня управління програмно-конфігурованої мережі.

5. Перелік завдань, які потрібно розробити:
- провести аналітичний огляд інфраструктур рівня управління програмно-конфігуровних мереж:
- розробити власний спосіб побудови рівня-управління, який зменшить час відгуку контролера

6. Перелік ілюстративного матеріалу: визначення основних показників в модифікованій структурі, визначення основних показників в централізованій структурі, визначення основних показників в децентралізованій локальній

структурі, визначення основних показників в децентралізованій глобальній структурі, визначення основних показників в ієрархічній структурі, час відгуку контролера в різних структурах рівня управління.

7. Перелік публікацій:
- тези доповіді "Мобільна пошукова система для динамічного контенту ", ПМК, 2016;
- тези доповіді " Порівняння масштабованості різних типів рівня управління в програмно-конфігурованих мережах", ПМК, 2018;
- стаття "Математичне порівняння продуктивності топологій рівня управління програмно-конфігуровних мереж ", Інтернаука.

8. Дата видачі завдання 5 вересня 2016 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ґрунтовне ознайомлення з предметною галуззю	21.11.2016	
2.	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури, патентний пошук	11.04.2017	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	27.05.2017	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації	04.10.2017	
5.	Проведення наукового дослідження; робота над третім розділом магістерської дисертації	11.12.2017	
6.	Проведення наукового дослідження; підготовка матеріалів доповіді на конференції ПМК-2018	10.02.2018	
7.	Завершення роботи над основною частиною магістерської дисертації; підготовка ілюстративного матеріалу	05.04.2018	
8.	Оформлення текстової і графічної частини магістерської дисертації	20.04.2018	
9	Попередній розгляд магістерської дисертації на кафедрі	26.04.2018	

Студент _____ _____
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ _____
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Актуальність теми. З розвитком Інтернету контроль мережі став вузьким місцем в класичному мережевому підході. Це спричинило появу нової програмно-конфігуровної архітектури SDN (Software defined network), в якій рівень управління мережею відділений від пристроїв передачі даних і реалізується програмно. SDN розділяє рівень контролю і рівень даних, що призводить до значного спрощення модуля управління. Таким чином, вся логіка управління мережею зосереджена, наприклад, на одному сервері, а інші пристрої мережі, відповідно до концепції, використовуються тільки для передачі даних. Проте одним із основних недоліків нової архітектури є складність масштабування рівня контролю мережею. Зі збільшенням кількості вузлів у мережі, і, відповідно, зі значним збільшенням запитів до рівня управління, навантаження на модуль контролю зростає, і його продуктивність може значною мірою знижуватися. Таким чином, обчислення масштабованості рівня управління SDN є критичною проблемою для успішної адаптації SDN до великомасштабних мереж або мереж з великою кількістю потоків.

Об'єктом дослідження є програмно-конфігуровні мережі.

Предметом дослідження є спосіб побудови рівня управління програмно-конфігуровної мережі.

Мета роботи: підвищити ефективність функціонування програмно-конфігуровних мереж за рахунок модифікації структури рівня контролю.

Наукова новизна полягає в наступному:

1. Проведено аналітичний огляд інфраструктур рівня управління програмно-конфігуровних мереж SDN та показано, що на сьогодні є невирішеним питання масштабованості і швидкодії контролерів в мережі.

2. Запропоновано спосіб побудови рівня управління мережі SDN, який відрізняється від існуючих швидкодією та дозволяє зменшити час відгуку контролера приблизно в 5 разів.

Практична цінність отриманих в роботі результатів полягає в тому, що запропонована структура призводить до можливості збільшення кількості вузлів в мережі без втрати продуктивності програмно-конфігурованої мережі, порівняно з існуючими структурами.

Апробація роботи. Основні положення і результати роботи були представлені та обговорювались на X науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2018 (Київ, 21-23 березня 2018 р.) та VIII науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2016 (Київ, 20–22 квітня 2016 р.).

Публікації. За матеріалами проведених досліджень опубліковано 3 наукових праці, з яких одна стаття та дві тези доповіді на конференції.

Структура та обсяг роботи. Магістерська дисертація складається з вступу, трьох розділів та висновків.

У вступі подано загальну характеристику роботи, зроблено оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень, сформульовано мету і задачі досліджень, показано наукову новизну отриманих результатів і практичну цінність роботи, наведено відомості про апробацію результатів і їхнє впровадження.

У першому розділі розглянуто опис предметної області досліджень та було наведено обґрунтування теми магістерської роботи.

У другому розділі наведено опис базових структур програмно-конфігурованих мереж.

У третьому розділі описано покращену структуру програмно-конфігурованої мережі та проведено порівняння її з існуючими.

У висновках представлені результати проведеної роботи.

Робота представлена на 80 аркушах, містить посилання на список використаних літературних джерел.

Ключові слова: програмно-конфігуровні мережі, масштабованість мережі, рівень контролю, час відгуку контролера, OpenFlow.

ABSTRACT

Actuality of theme. With the development of the Internet, network monitoring has become a bottleneck in the classic network approach. This led to the emergence of a new software architecture SDN (Software Defined Network), where the level of network management is separated from the data transfer devices and implemented programmatically. The SDN separates the monitoring level and the data layer, which results in a significant simplification of the control module. Thus, the whole network management logic is concentrated, for example, on one server, and other network devices, according to the concept, are used only for data transmission. However, one of the main disadvantages of the new architecture is the complexity of scaling the level of network control. With the increase in the number of nodes in the network, and, accordingly, with a significant increase in requests to the level of control, the load on the control module grows and its productivity can be significantly reduced. Thus, computing the scalability of the SDN management layer is a critical issue for the successful adaptation of SDN to large-scale networks or networks with a large number.

The object of the study is a software defined network.

The subject of the study is a method of constructing the control plane software defined network.

Purpose: to increase the efficiency of functioning of software defined networks by modifying the structure of the level of control.

The scientific novelty is

1. An analytical review of the management-level infrastructures of the software-defined SDN networks was conducted and it was shown that today the question of scalability and speed of controllers in the network is unresolved.

2. A method for constructing the management level of the SDN network is proposed, which differs from the existing ones by the speed and allows reducing the response time of the controller by about 5 times.

The practical value of the results obtained in the work is that the proposed structure leads to the possibility of increasing the number of nodes in the network without losing the performance of software-configurable networks in comparison with existing structures..

Approbation. The main provisions and results of the work were presented and discussed at the Xth scientific conference of undergraduates and postgraduates "Applied Mathematics and Computing" PMK-2018 (Kiev, March 21-23, 2018) and the VIII Scientific Conference of Master and Postgraduate students "Applied Mathematics and Computing" PMK-2016 (Kiev, April 20-22, 2016).

Publications. Based on the materials of the research, 3 scientific papers have been published, one of which is article and two theses of the report at the conference.

Structure and scope of work. The master's thesis consists of an introduction, three chapters and conclusions.

The introduction presents a general description of the work, an assessment of the current state of the problem is made, the relevance of the research direction is substantiated, the goals and objectives of the research are stated, the scientific novelty of the results obtained and the practical value of the work are shown, information is provided on the approbation of the results and their implementation.

The first section describe the subject area of the studies is presented and the reasons for the master's work have been given.

The second section describes the basic structures of software defined networks.

The third section describes the improved structure software defined networks and compares it with existing ones.

The conclusions contain the results of the work done.

The work is presented on 82 pages, contains references to a list of literary sources used.

Keywords: software defined network, network scalability, control plane, controller RTT, OpenFlow.

РЕФЕРАТ

Актуальность темы. С развитием Интернета контроль сети стал узким местом в классическом сетевом подходе. Это привело к появлению новой программно определенной архитектуры SDN (Software defined network), в которой уровень управления сетью отделен от устройств передачи данных и реализуется программно. SDN разделяет уровень контроля и уровень данных, что приводит к значительному упрощению модуля управления. Таким образом, вся логика управления сетью сосредоточена, например, на одном сервере, а другие устройства сети, согласно концепции, используются только для передачи данных. Однако одним из основных недостатков новой архитектуры является сложность масштабирования уровня контроля сети. С увеличением количества узлов в сети, и, соответственно, со значительным увеличением запросов к уровню управления, нагрузка на модуль контроля растет, и его производительность может значительно снижаться. Таким образом, вычисление масштабируемости уровня управления SDN является критической проблемой для успешной адаптации SDN к крупномасштабным сетям или сетям с большим количеством потоков.

Объектом исследования является программно определенные сети.

Предметом исследования является способ построения уровня управления программно определенной сети.

Цель работы: повысить эффективность функционирования программно определенных сетей за счет модификации структуры уровня контроля.

Научная новизна заключается в следующем:

1. Проведен аналитический обзор инфраструктур уровня управления программно определенных сетей SDN и показано, что на сегодня является нерешенным вопрос масштабируемости и быстродействия контроллеров в сети.

2. Предложен способ построения уровня управления сети SDN, который отличается от существующих быстродействием и позволяет уменьшить время отклика контроллера примерно в 5 раз.

Практическая ценность полученных в работе результатов заключается в том, что предложенная структура приводит к возможности увеличения количества узлов в сети без потери производительности программно-конфигурируемых сети по сравнению с существующими структурами.

Апробация работы. Основные положения и результаты работы были представлены и обсуждались на X научной конференции магистрантов и аспирантов «Прикладная математика и компьютинг» ПМК-2018 (Киев, 21-23 марта 2018) и VIII научной конференции магистрантов и аспирантов «Прикладная математика и компьютинг» ПМК-2016 (Киев, 20-22 апреля 2016).

Публикации. По материалам проведенных исследований опубликовано 3 научных труда, из которых одна статья и две тезисы доклада на конференции.

Структура и объем работы. Магистерская диссертация состоит из введения, трех глав и выводов.

Во *введении* представлена общая характеристика работы, произведена оценка современного состояния проблемы, обоснована актуальность направления исследований, сформулированы цели и задачи исследований, показано научную новизну полученных результатов и практическую ценность работы, приведены сведения об апробации результатов и их внедрение.

В *первом разделе* рассмотрены описание предметной области исследований и были приведены обоснования темы магистерской работы.

Во *втором разделе* описаны базовых структур программно-конфигурируемых сетей.

В *третьем разделе* описано улучшенную структуру программно-конфигурируемых сети и проведено сравнение ее с существующими.

В *выводах* представлены результаты проведенной работы.

Работа представлена на 82 листах, содержит ссылки на список использованных литературных источников.

Ключевые слова: программно определенный сети, масштабируемость сети, уровень контроля, время отклика контроллера, OpenFlow.

ЗМІСТ

СПИСОК ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

API	Application programming interface
AT	Asynchronous Transfer Mode
CLI	Command-line interface
ETSI	European Telecommunications Standards Institute
KVM	Kernel-based Virtual Machine
MTD	Mobile network
NBI	Northbound interface
NVF	Network virtualization function
OVS	Open vSwitch
OVSDB	Open vSwitch Database
RGDD	Reliable Group Data Delivery
SDK	Service-level agreement
SDN	Software Defined Network
SD-WAN	Software development kit
SLA	Software-Defined WAN
SPAN	Switched Port Analyzer
VM	Virtual Machine
WAN	Wide area network
XAPI	Experience API

ВСТУП

У зв'язку з постійним розвитком інформаційних технологій і появою нових технологій (таких як хмарні обчислення та Big Data), вимоги до комп'ютерних мереж зростають, а реалізація ускладнюється. Мережі потребують більшої швидкості передачі даних та вдосконалення інструментів, що використовуються для мережевого управління і моніторингу. Така ситуація призводить до появи нових функціональних і технологічних мереж, з ускладненою інфроструктурою. Старі методи моніторингу і управління не відповідають новим вимогам.

Тому останнім часом зростає популярність программно-конфігуровних мереж SDN (Software-Defined Networks). Самій ідеї мереж SDN вже більше десяти років, але в останні декілька років відомі компанії пропонують нові реалізації, які відкривають більше можливостей. Одною з найпопулярніших є організація мережі SDN зі спільним застосуванням протоколу OpenFlow. Головна перевага представленої технології в тому, що вона працює окремо від мережевих пристроїв і її контроль може здійснюватися операторами за допомогою стандартного сервера.

Проте, як і кожна нова технологія, SDN досі має багато проблем, вирішення яких є актуальною проблемою. Основними є масштабованість і безпека рівня управління.

1. 1. КОНЦЕПЦІЯ ТА ОСОБЛИВОСТІ ПОБУДОВИ SDN

1.1 Загальна концепція програмно-конфігуровних мереж і причини виникнення

Загальний опис

Головна мета SDN є відокремлення рівня додатків від рівня управління і рівня управління від рівня передачі даних. Таким чином, можна значно спростити складність фізичних пристроїв, оскільки виконування логічних функції повністю переноситься на вищий рівень. Це не тільки здешевлює фізичні пристрої, а й покращує надійність і спрощує управління мережі в цілому. Тепер, замість маршрутизаторів можна використовувати звичайні комутатори. Для адміністратора мережі буде значно легше контролювати мережу в цілому. Також, це дозволяє абстрагуватися від реалізації кожного конкретного пристрою, оскільки рівень управління зв'язується з рівнем даних через стандартний інтерфейс. А це, в свою чергу, значно спрощує взаємодію між пристроями різних виробників, і зменшує час налаштування і підготовки або ремонту всієї мережі.

Окрім того, така побудова мережі значно прискорює створення нових мережевих додатків. Оскільки для взаємодії рівнів додатків і управління також використовується стандартний інтерфейс, програмісту більше не потрібно думати про те, яким саме чином можна передавати команди і запити до мережі. Головне реалізувати основний функціонал додатку, а додаток буде взаємодіяти з рівнем контролю через попередньо виділені інтерфейси. Архітектура SDN схематично зображена на рисунку 1.1. В таблиці 1.1 вказано основні відмінності SDN від традиційного підходу.

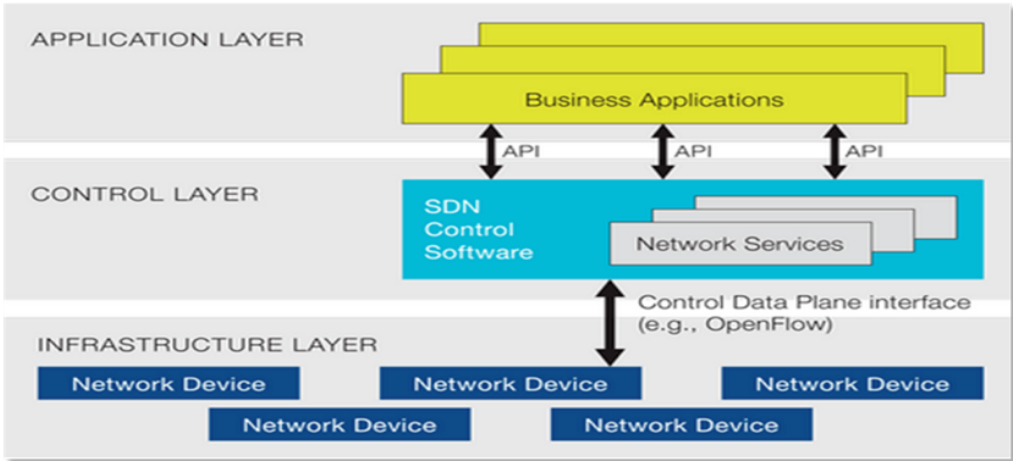


Рисунок 1.1 - Схематичне зображення архітектури SDN

Таблиця 1.1. - Порівняння підходу SDN з традиційними мережами

	Традиційні мережі	SDN
Мережевий вигляд	За рахунок апаратного забезпечення	За рахунок програмного забезпечення
Контроль за конфігурацією	Постачальник апаратного забезпечення	Користувач
Відкритість мережі	Закрита структура	Відкрита структура
Сумісність	Незалежні протоколи	Стандартизовані протоколи
Управління мережею	Управління на низькому рівні	Управління на логічному рівні
Адоптація нових технологій	Відповідно до потреб постачальника	Відповідно до потреб користувача

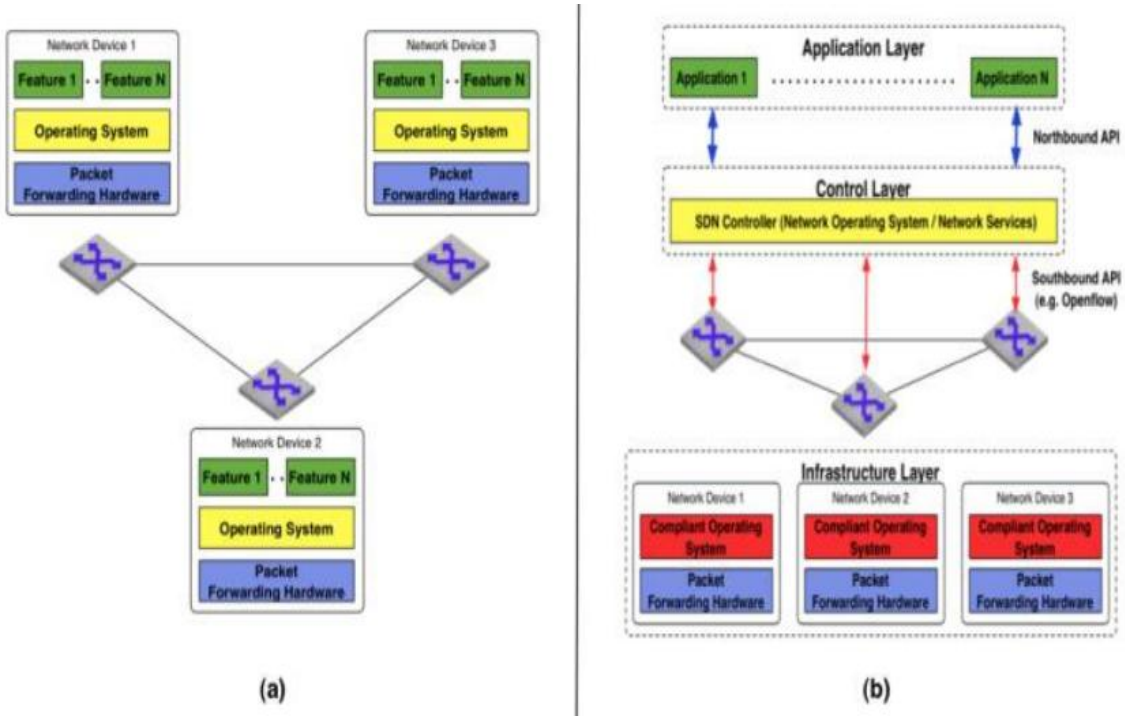


Рисунок 1.2 - Традиційна архітектура (a) і архітектура SDN (b)

На рисунку 1.2 можна побачити різницю між традиційною архітектурою і архітектурою SDN.

З SDN додатки отримують представлення про мережу, на відміну від традиційних мереж, де мережа отримує представлення про додаток:

- традиційні (тобто не SDN) додатки лише неявно та опосередковано описують свої вимоги до мережі, зазвичай включають декілька кроків що обробляються людьми, наприклад, перевірка наявності ресурсів та керування політикою конфіденційності для підтримки додатків;
- традиційні мережі (наприклад, сучасний Інтернет та його послуги, такі як веб-переглядач, потокове передавання мультимедіа) не пропонують динамічний спосіб виразити весь спектр вимог користувачів, наприклад пропускну спроможність, затримку, варіацію затримки або доступність. Пакетні заголовки можуть зашифрувати запити пріоритету, але постачальники мережі зазвичай не довіряють маркуванням користувацького трафіку. Тому деякі мережі намагаються самостійно вивести вимоги користувачів (наприклад, через аналіз трафіку), що може спричинити додаткові витрати, а іноді й призвести

до неправильної класифікації. SDN надає користувачеві можливість повністю вказувати свої потреби в контексті довірених відносин;

- традиційні (тобто не SDN) мережі не надають інформацію та стан мережі додаткам, що їх використовують. За допомогою підходу SDN додаток може відслідковувати стан мережі та адаптувати його відповідно.

Рівень управління логічно централізований і відокремлений від рівня даних. Контролер SDN підсумовує інформацію про стан мережі для додатків і переводить вимоги додатків до низького рівня.

- Це не означає, що контролер фізично є централізованим. З точки зору продуктивності, масштабованості та/або надійності логічно централізований контролер SDN може бути розподілений таким чином, щоб декілька екземплярів фізичного контролера співпрацювали для керування мережею та обслуговування додатків.
- Контрольні рішення приймаються на глобальному перегляді стан мережі, а не розподіляються по ізольованій поведінці в кожній підмережі. За допомогою SDN рівень керування виступає в ролі єдиної логічно централізованої мережевої операційної системи з точки зору як планування, так і вирішення конфліктів ресурсів, а також абстрагування від деталей пристроїв низького рівня, наприклад, від електричної або оптичної передачі.

Контролер SDN має повний контроль над рівнем даних SDN, з огляду на обмеження його можливостей, і таким чином не повинен конкурувати з іншими елементами рівня контроля, що спрощує планування та розподіл ресурсів. Це дозволяє мережам працювати з більш складними та більш точними політиками з використанням мережевих ресурсів та гарантією якості послуг. Це відбувається через загальну інформаційну модель (наприклад, як визначено OpenFlow).

SDN дозволяє програмувати поведінку мережі в централізованому режимі через програмні додатки, що використовують відкриті API.

Впровадження загального рівня управління SDN дозволяє керувати всією мережею та її пристроями послідовно, незалежно від складності базової мережевої технології.

SDN дозволяє послідовно керувати мережею, яка складається з технологічно складних частин.

Атрибути відкритості в SDN

Відкриті стандарти: вільно та загальнодоступні специфікації для апаратного або програмного забезпечення, розроблені та підтримуються за допомогою спільних зусиль користувачів і постачальників обладнення.

Програмне забезпечення з відкритим кодом: вихідний код доступний для будь-кого для зміни або покращення, як OpenStack та OpenDaylight.

API та SDK: API слугують інструментом для створення програмних додатків, часто визначаючи, яким чином компоненти програм повинні взаємодіяти один з одним. SDK - це пакети попередньо написаного коду, що мінімізує кількість повторної розробки коду. Деякі SDK публікуються API, доступні для всіх для написання у власних програмах

Відкрите обладнання: проекти з відкритим посиланням на обчислювальні та мережеві продукти, такі як Open Compute Project

Основоположними технологіями є SDN контролери, vSwitches або API-програми. Вони дотримуються більш високих стандартів, ніж підтримуючі технології, які дозволяють іншим технологіям взаємодіяти.

Основні переваги SDN полягають в наступному.

1) Можливість програмувати мережу: SDN дозволяє керувати поведінкою мережі програмним забезпеченням, яке знаходиться поза мережевими пристроями, що забезпечують фізичний зв'язок. Як наслідок, з'являється можливість налаштувати поведінку своїх мереж для підтримки нових послуг і навіть окремих клієнтів. Від'єднання апаратного забезпечення та програмного дозволяє запровадити інноваційні, диференційовані нові служби швидко, без обмежень закритих платформ.

2) Логічно централізований контроль: SDN побудований на мережевих топологіях, які дозволяють централізовано керувати мережевими ресурсами. Розподіляються традиційні методи керування мережею. Пристрої функціонують автономно. Завдяки такому централізованому управлінню забезпечується управління пропускною спроможністю, відновлення та безпека. А адміністратор отримує цілісний погляд на мережу.

3) Абстракція мережі: сервіси та додатки, що працюють на технології SDN, абстрагуються від низькорівневих технологій і апаратного забезпечення. Програми взаємодіють з мережею через API замість інтерфейсів керування, які тісно пов'язані з апаратним забезпеченням.

4) Відкритість: архітектура SDN відкриває нову еру відкритості, що забезпечує можливість взаємодії багатьох виробників, а також сприяє створенню нейтральної інфраструктури для взаємодії постачальників послуг. Відкриті API підтримують широкий спектр додатків, включаючи хмарні послуги, OSS / BSS, SaaS і важливі мережеві додатки. Крім того, програмне забезпечення може контролювати обладнання від декількох постачальників з відкритими програмними інтерфейсами, такими як OpenFlow. Окрім того, сервіси мережі та програми можуть працювати в межах спільного програмного середовища.

Ключовою перевагою технології SDN є здатність операторів мережі створювати програми, які використовують API SDN, і надає додаткам можливість контролювати поведінку мережі. SDN дозволяє користувачам розробляти мережеві програми, спостерігати за вимогами мережі та автоматично адаптувати конфігурацію мережі, якщо необхідно.

Програмне забезпечення, визначене SDN, є новим підходом до хмарних обчислень, що полегшує управління мережею та забезпечує програмно ефективну конфігурацію мережі для поліпшення продуктивності мережі та моніторингу. Мережі SDN призначені для вирішення того, що статична архітектура традиційних мереж децентралізована та складна, тоді як існуючі мережі вимагають більшої гнучкості та більше легкого вирішення проблем.

SDN вміщає рівень управління мережею в одному компоненті, розділивши процес перенаправлення мережевих пакетів і процесу маршрутизації. Рівень управління складається з одного або більше контролерів, які розглядаються як мозок мережі SDN, де інтегрована вся логіка. Проте централізація має свої недоліки щодо безпеки, масштабованості та еластичності, і це є головною проблемою SDN.

Мережі SDN зазвичай пов'язують з протоколом OpenFlow (для віддаленого зв'язку з елементами мережевого рівня з метою визначення шляху мережевих пакетів через мережеві комутатори) з моменту появи останньої в 2011 році. З 2012 року багато компаній відійшли від OpenFlow і прийняли різні методи. До них належать Cisco Systems Open Network Environment та мережа віртуалізації мережі Nicira. Схема використання протоколу OpenFlow зображена на рисунку 1.3.

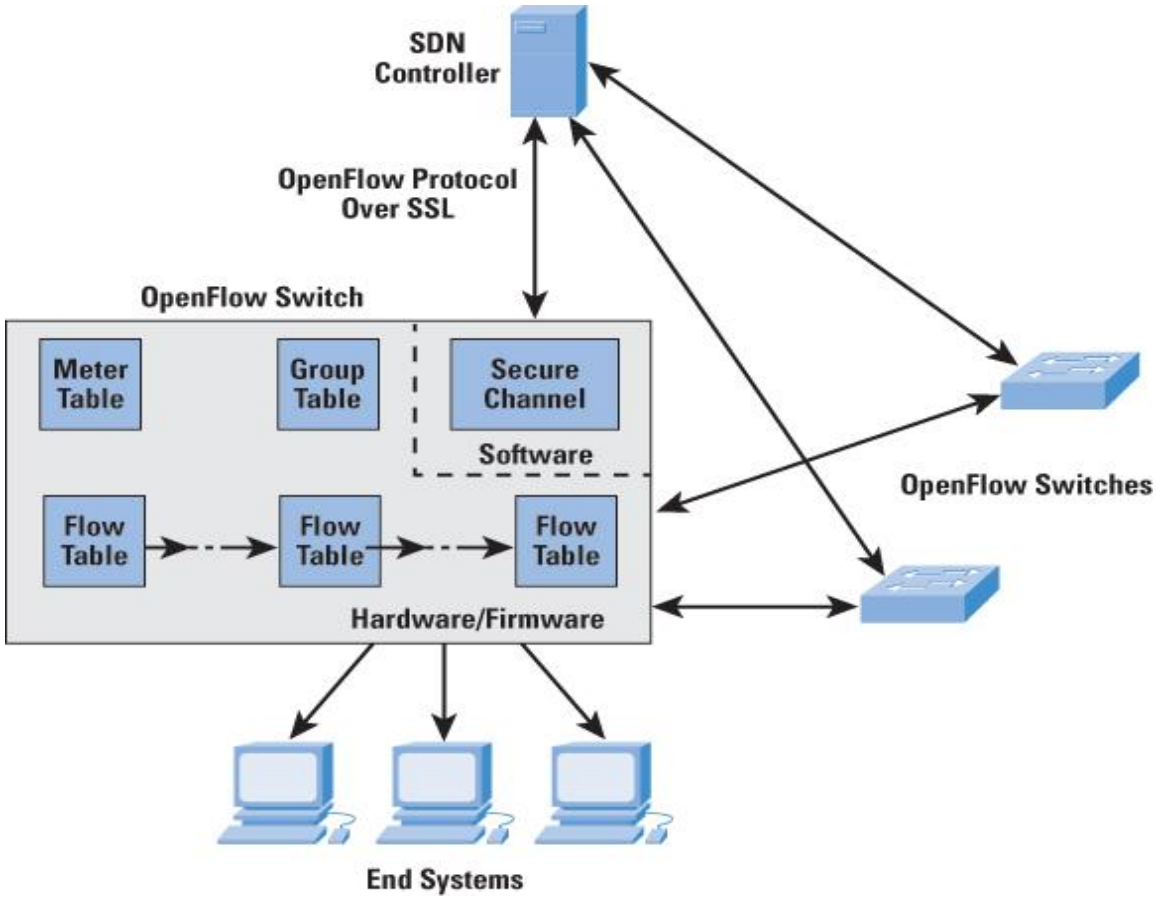


Рисунок 1.3 - Використання протоколу OpenFlow в мережі SDN

SD-WAN застосовує подібну технологію до широкосмугової мережі (WAN).

Історію SDN можна простежити до відокремлення лінії керування та даних, яка спочатку використовувалася в комутованій телефонній мережі загального користування, як спосіб спрощення надання та керування набагато раніше, ніж ця архітектура стала використовуватись у мережах передачі даних.

Комітет IETF почав розглядати різні способи відокремлення функції управління та передачі в запропонованому стандарті інтерфейсів, опублікованому в 2004 році та названим «Розділ для пересилання та управління елементами» (ForCES). Робоча група ForCES також запропонувала Companion SoftRouter Architecture. Додаткові стандарти від IETF, які передбачали відокремлення управління від даних, включають Linux Netlink як протокол IP-послуг та архітектуру на основі елемента обчислення шляху (PCE).

Ці ранні спроби не були успішними з двох причин. Одна з них полягає в тому, що відокремлення управління від даних може бути ризикованими, особливо через потенційні помилки на рівні управління. Друга полягає в тому, що постачальники створювали прикладні інтерфейси (API), що не взаємодіяли між собою.

SDN - це динамічна, керована, економічно ефективна та адаптована архітектура, придатна для високошвидкісної, динамічної природи сучасних програм. Архітектура SDN відокремлює функції керування та направлення трафіку в мережі, що дозволяє керувати базовою інфраструктурою, яка повинна абстрагуватися від додатків та сервісів мережі.

Інфраструктурний рівень складається з мережевих елементів, які виявляють свої можливості відносно рівня контроля. Додатки SDN існують на прикладному рівні та передають свої вимоги до мережі до рівня управління через північний інтерфейс, що називають NBI. Рівень контролю SDN передає вимоги додатків на низький рівень і контролює мережеві елементи. SDN може поєднувати конкуруючі вимоги додатків для обмежених мережевих ресурсів відповідно до політики конфіденційності.

1.2 Архітектура SDN

Основні особливості перелічені далі.

Мережеве керування повністю програмне, оскільки воно відокремлене від функцій пересилання.

Абстрактний контроль над пересиланням дозволяє адміністраторам динамічно коригувати потоки трафіку всередині мережі, щоб задовольнити тимчасові потреби.

Програмні контролери SDN підтримують глобальний погляд на мережу.

SDN дозволяє операторам мережі налаштовувати, керувати, захистити та оптимізувати ресурси мережі за допомогою динамічних автоматизованих SDN-програм, які вони можуть створювати самостійно, оскільки програми не залежать від фірмового програмного забезпечення.

Відкритий стандарт та нейтральний постачальник. При реалізації за допомогою відкритих стандартів SDN спрощує проектування та експлуатацію мережі, оскільки інструкції надаються контролерами SDN замість декількох пристроїв і протоколів, специфічних для постачальників.

Велика популярність мобільних пристроїв і контенту, віртуалізація серверів та поява хмарних сервісів є ключовими тенденціями, що стимулюють мережеву індустрію переглянути традиційні мережеві архітектури. Багато традиційних мереж є ієрархічними, побудованими за допомогою ярусів, комутаторів Ethernet, розташованих у структурі дерева. Ця конструкція мала сенс, коли клієнт-серверні обчислення були домінуючими, але така статична архітектура не підходить для потреб динамічних обчислень та зберігання даних сьогоднішніх корпоративних центрів обробки даних.

Деякі сучасні технічні тенденції, що викликають потребу в новій парадигмі мережі, включають наступне.

Зміна шаблонів трафіку

На відміну від клієнт-серверних додатків, де основна частина зв'язку відбувається між клієнтами і одним сервером, сьогоднішні програми отримують доступ до декількох баз даних і серверів, створюючи велику кількість трафіку. У той же час користувачі змінюють мережеві схеми трафіку, коли вони намагаються в будь-який час отримати доступ до корпоративного вмісту та програм від будь-якого типу пристрою (включаючи їх власні), підключаючись з будь-якого місця. Нарешті, багато менеджерів корпоративних центрів обробки даних розглядають модель корисної обчислювальної системи, яка може включати в себе приватну хмара, загальну хмару або сукупність обох, що призведе до додаткового трафіку в широкосмуговій мережі.

Узагальнення стандартів IT

Користувачі все частіше використовують мобільні персональні пристрої, такі як смартфони, планшети та ноутбуки для доступу до мережі. З'являється проблема, як пристосувати ці особисті пристрої та, водночас, захистити корпоративні дані та інтелектуальну власність.

Розвиток хмарних сервісів

Користувачам необхідно мати оперативний доступ до додатків, інфраструктури та інших інформаційних ресурсів. Окрім того, планування інфраструктури для хмарних сервісів повинно здійснюватися в умовах підвищеної безпеки, відповідності вимогам аудиту та реорганізації бізнесу, консолідації та злиття. Забезпечення послуг самообслуговування, як приватної, так і загальнодоступної хмари, вимагає еластичного масштабування обчислень, сховища та ресурсів мережі. Прогноз розвитку популярності хмарних обчислень можна побачити на рисунку 1.4.

Big Data потребує більшу пропускну спроможність

Використання сьогодні Big Data або великих наборів даних потребує масової паралельної обробки даних, що розподіляється на тисячі серверів, для чого потрібні прямі зв'язки кожного сервера один з одним.

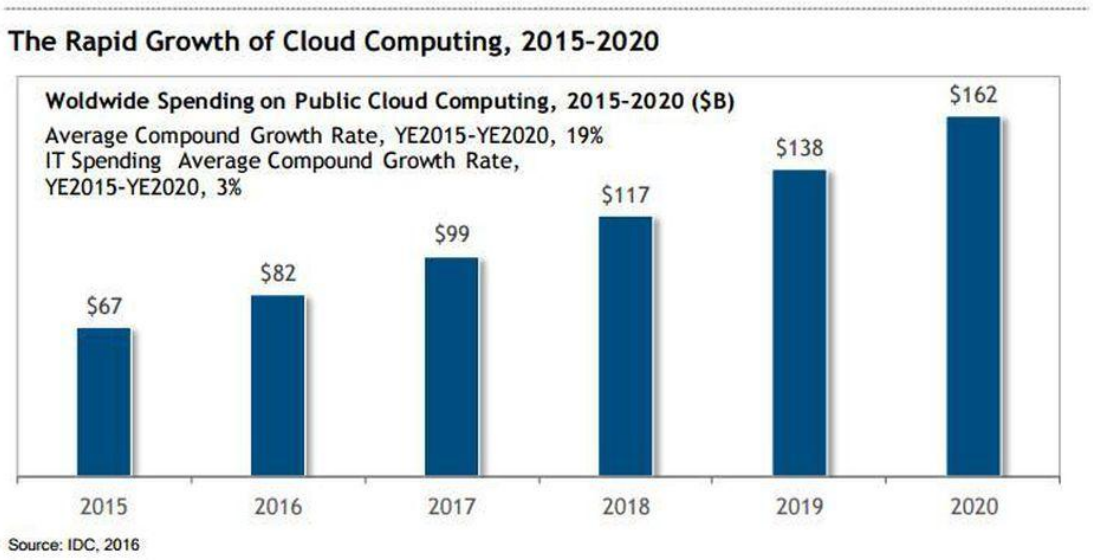


Рисунок 1.4 - Прогноз інвестиції в розвиток хмарних сервісів

Збільшення кількості наборів даних вимагає постійного збільшення масштабованості і продуктивності центрів обробки даних. Оператори мереж великомасштабних центрів обробки даних зустрічаються з проблемою масштабування мережі значних розмірів, оскільки необхідно уникати розривів з'єднань.

Наступний список визначає та пояснює архітектуру SDN

Рівень додатків

Мережеві додатки - це додатки, які програмно передають свої вимоги до мережі та бажану поведінку мережі до контролера SDN через NBI (northbound interface). Крім того, вони можуть використовувати абстрактний вид мережі для внутрішнього прийняття рішень. Додаток SDN складається з логічної частини та одного або декількох драйверів NBI. Додатки SDN можуть додавати шар абстрактного контроля мережею, таким чином, пропонуючи один або декілька NBI вищого рівня через відповідних агентів NBI.

Рівень управління

Контролер є логічно централізованою сутністю, що відповідає за переведення вимог із рівня додатків SDN до рівня фізичної мережі SDN. Також контролер SDN відповідає за надання додаткам абстрактного подання мережі (що може включати статистичні дані та події). Контролер SDN складається з одного або декількох агентів NBI, логіки управління SDN та драйвера управління контрольним інтерфейсом (CDPI).

Рівень передачі даних

Фізична інфроструктура SDN - це сукупність мережевих пристроїв, що забезпечують видимість в мережі та безпосереднє керування пересиланням та обробкою даних. Логічне представлення може охоплювати як всі ресурси мережі, так і невелику підмножину фізичних ресурсів. Фізична мережа складається з агента CDPI, з одного або декількох ядер для переадресації трафіку, а також може використовувати функції обробки трафіку. Ці ядра та функції можуть включати в себе просте пересилання між зовнішніми інтерфейсами або функціями внутрішньої обробки трафіку, або функціями завершення.

SDN-контроль до плану даних інтерфейсу (CDPI)

SDN CDPI - це інтерфейс, визначений між контролером і фізичною мережею, який забезпечує щонайменше (i) програмне управління усіма операціями пересилання; (ii) оголошення можливостей; (iii) звітування про статистику; (iv) повідомлення про події. Одна з основних ідей SDN полягає в тому, що CDPI реалізується відкритим, нейтральним та сумісним способом.

SDN Northbound Interfaces (NBI)

SDN NBI є інтерфейсами між SDN-додатками та SDN-контролерами, і, як правило, забезпечують абстрактне представлення мережі та дозволяють прямо описувати поведінку та вимоги до мережі. Це може відбутися на будь-якому рівні абстракції і через різні набори функціональності. Ці інтерфейси реалізуються відкритим, нейтральним та сумісним способом.

Місце інтерфейсів в загальній архітектурі SDN представлено на рисунку 1.5.

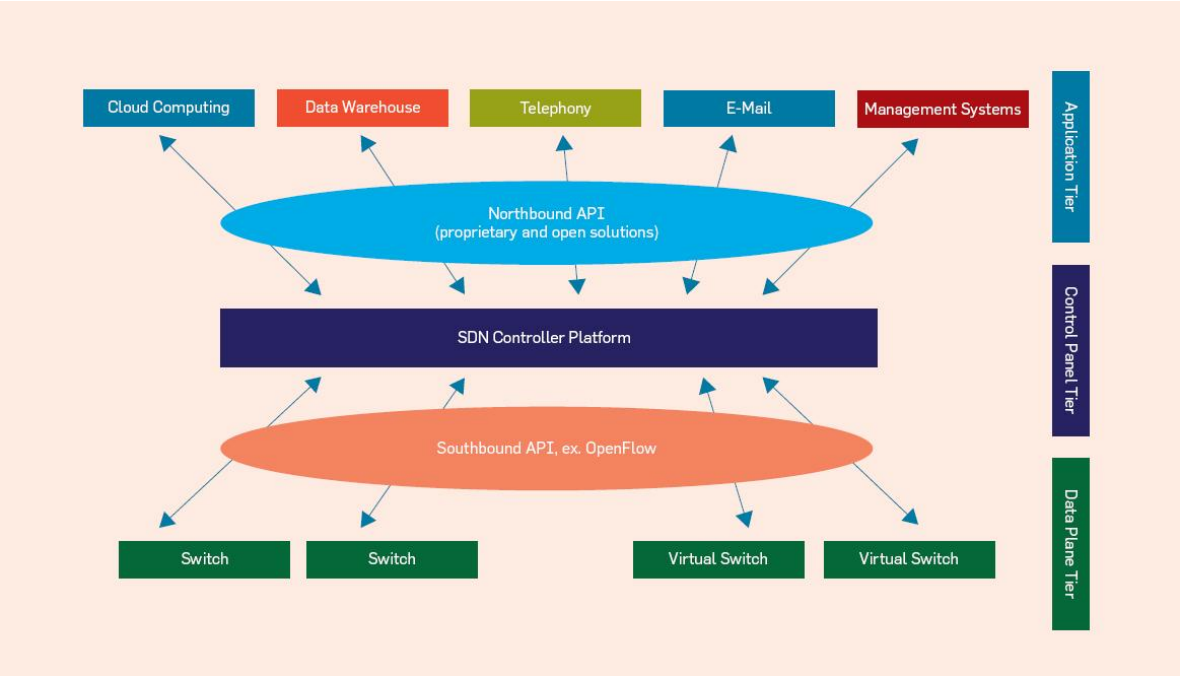


Рисунок 1.5 - Представлення функцій інтерфейсів

Рівень управління SDN

Централізований - ієрархічний - розподілений

Архітектура рівня управління SDN може бути спроектована централізованою, ієрархічною або децентралізованою. В перших реалізаціях рівень управління SDN був реалізований за централізованою структурою, де єдиний контрольний орган має глобальний погляд на мережу. Хоча це спрощує реалізацію логіки управління, вона має обмеження на масштабованість, оскільки розмір і динаміка трафіку в мережі збільшуються. Щоб подолати ці обмеження, запропоновано декілька підходів, які поділяються на дві категорії: ієрархічна та повністю розподілена. У ієрархічних структурах розподілені контролери працюють в режимі роздільної мережі, тоді як рішення, які потребують глобальний вид мережі, приймаються логічно централізованим кореневим контролером. У

розподілених структурах контролери працюють за своїм локальним представленням або вони можуть обмінюватися повідомленнями синхронізації, щоб збільшити видимість мережі. Розподілені рішення більш підходять для підтримки адаптивних SDN-додатків.

Розміщення контролера

Ключовим питанням при розробці розподіленого рівня управління SDN є прийняття рішення щодо кількості та розміщення контрольних модулів. Важливим параметром для розгляду при цьому є затримка розповсюдження між контролерами та мережевими пристроями, особливо в контексті великих мереж. Окрім того, багато уваги приділяється надійності керування, відмовостійкості та вимогам до застосування.

Передача потоку SDN

OpenFlow використовує TCAM-таблиці для маршрутизації пакетів (потоків). Якщо потоки приходять до комутатора, виконується пошук таблиці маршрутизації. Залежно від реалізації таблиці маршрутизації, це робиться в таблиці маршрутизації програмного забезпечення, якщо використовується vSwitch або в ASIC, якщо він реалізований у апаратному забезпеченні. У випадку, коли не знайдено відповідний потік, надсилається запит до контролера для подальших інструкцій. Він обробляється в одному з трьох різних режимів. У реактивному режимі контролер починає роботу після цих запитів і створює та встановлює правило у таблиці маршрутизації для відповідного пакета, якщо це необхідно. У проактивному режимі контролер заповнює таблицю потоків записами для всіх можливих варіантів трафіку, які можливі для цього комутатора, заздалегідь. Цей режим можна порівняти з типовими елементами таблиці маршрутизації сьогодні, де всі статичні записи встановлюються заздалегідь. Після цього ніякий запит не надсилається контролеру, оскільки всі вхідні потоки знайдуть відповідний запис. Основна перевага в проактивному режимі полягає в тому, що всі пакети перенаправляються за лінійною швидкістю (враховуючи всі записи поточних таблиць в TCAM), і ніяка затримка не додана. Третій спосіб, гібридний

режим, комбінує гнучкість реактивного режиму та переадресацію з низькою затримкою (проактивний режим) для трафіку [1].

Додатки

SDMN

Програмне забезпечення, розроблене для мобільних мереж (SDMN), є підходом до розробки мобільних мереж, де всі функціональні можливості протоколу реалізуються в програмному забезпеченні, максимально використовуючи загальні і спеціальні апаратні засоби та програмне забезпечення в базових мережах і мережах радіодоступу. Пропонується як

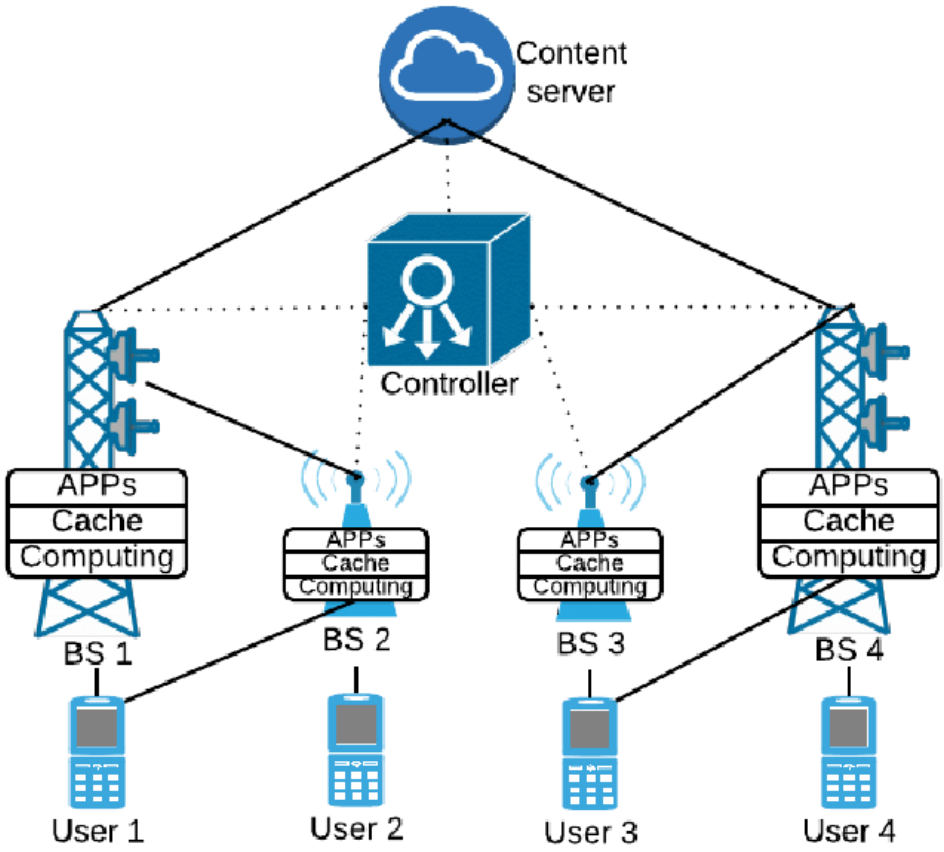


Рисунок 1.6 - Архітектура SDMN з підтримкою кеш-пам'яті з MEC

продовження парадигми SDN зі специфічними функціями мобільних мереж. Архітектура SDMN представлена на рисунку 1.6.

SD-WAN

SD-WAN - це широкосмугова мережа (WAN), за керування якою відповідає програмне забезпечення мережі. Основною перевагою SD-WAN є зниження витрат на WAN, використовуючи більш комерційно доступні орендовані лінії, як альтернативну або часткову заміну більш дорогих ліній MPLS.

Контроль та управління розділені окремо від апаратного забезпечення з центральними контролерами, що дозволяє простіше конфігурувати та адмініструвати мережу. На рисунках 1.7 і 1.8 представлена різниця між WAN і SD-WAN.

SD-LAN

SD-LAN є локальною мережею (LAN), що побудована за принципами програмно-конфігуровних мереж, хоча існують ключові відмінності в топології, безпеці мережі, видимості прикладних програм та контролі, керуванні та якості сервісу. SD-LAN відокремлює контроль за управлінням та дані для забезпечення архітектури, що відповідає вимогам локальної мережі, керованої політикою для дротових і бездротових локальних мереж. SD-LAN характеризуються використанням хмарної системи керування та бездротовим зв'язком без наявності фізичного контролера.

Безпека при використанні парадигми SDN

Архітектура SDN дозволяє спрощувати та покращувати програми, пов'язані з мережею, завдяки центральному поданню контролера мережею та можливості перепрограмувати рівень даних у будь-який час. Безпека самої архітектури SDN залишається відкритим питанням.

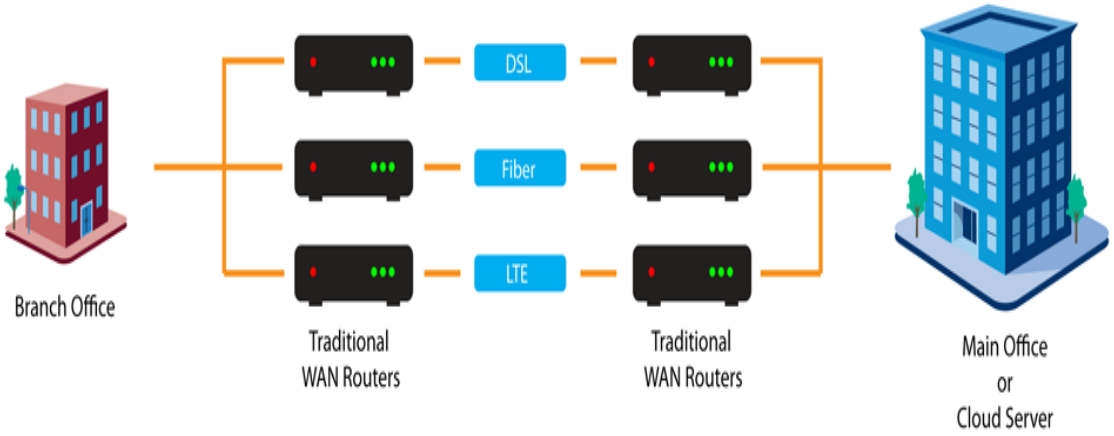


Рисунок 1.7 - Схематичне представлення традиційної WAN

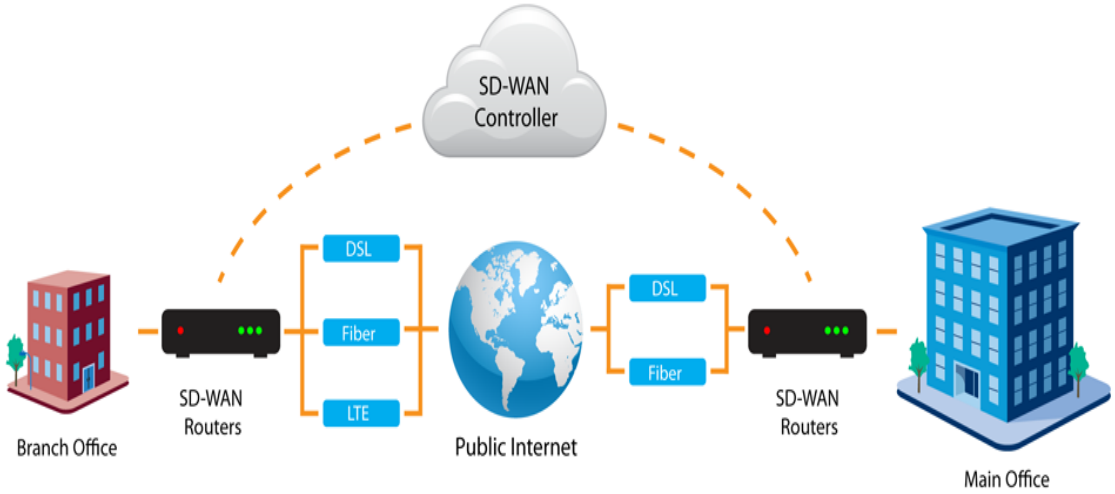


Рисунок 1.8 - Схематичне представлення SD-WAN

Існують дослідницькі роботи що торкаються питання безпеки в SDN, побудовані на контролері SDN, роблячи акцент на різних цілях. Деякі конкретні випадки використання таких застосувань - виявлення та пом'якшення розподіленої служби відмови (DDoS) та поширення вірусів - в основному ідея полягає в періодичному зборі статистики мережі з рівня передачі даних стандартним способом (наприклад, за допомогою Openflow). Після цього застосовується алгоритми класифікації цих статистичних даних для виявлення будь-яких мережевих аномалій. Якщо виявлено аномалію,

додаток наказує контролеру, як перепрограмувати лінію даних, щоб виправити помилки.

Інший вид додатка безпеки використовує контролер SDN шляхом впровадження деяких алгоритмів MTD. Алгоритми MTD, як правило, використовуються для того, щоб зробити будь-яку атаку на задану систему чи мережу складнішою, ніж зазвичай, періодично приховувати або змінювати ключові властивості цієї системи або мережі. У традиційних мережах реалізація алгоритмів MTD не є тривіальним завданням, оскільки складно побудувати центральний орган управління, здатний визначити, яку частину системи потрібно захищати, які ключові властивості приховані або змінені. У мережі SDN такі завдання стають більш простими завдяки центральному елементу контролера. Одна програма може, наприклад, періодично призначати віртуальні IP-адреси хостам у мережі, а з'єднання віртуальних IP-адрес з реальними IP адресами виконується контролером. Іншими вид додатків може імітувати підроблені відкриті/закриті/відфільтровані порти на випадкових вузлах мережі, щоб додати значний шум під час фази сканування мережі, здійсненого зловмисником.

Збільшити рівень безпеки в мережах, що підтримують SDN, можна використовуючи відповідно FlowVisor та FlowChecker. Приклад організації такої мережі зображений на рисунку 1.9.

Перший використовує єдине обладнання рівню даних, що поділене між декількома логічними мережами. Використовуючи цей підхід, ті самі апаратні ресурси можуть бути використані для виробництва та розробки, а також для розділення моніторингу, конфігурації та Інтернет-трафіку, де кожен сценарій може мати свою власну логічну топологію, яка називається фрагментом. У поєднанні з цим підходом FlowChecker реалізує перевірку нових правил OpenFlow, які розгортаються користувачами, що використовують власні фрагменти.

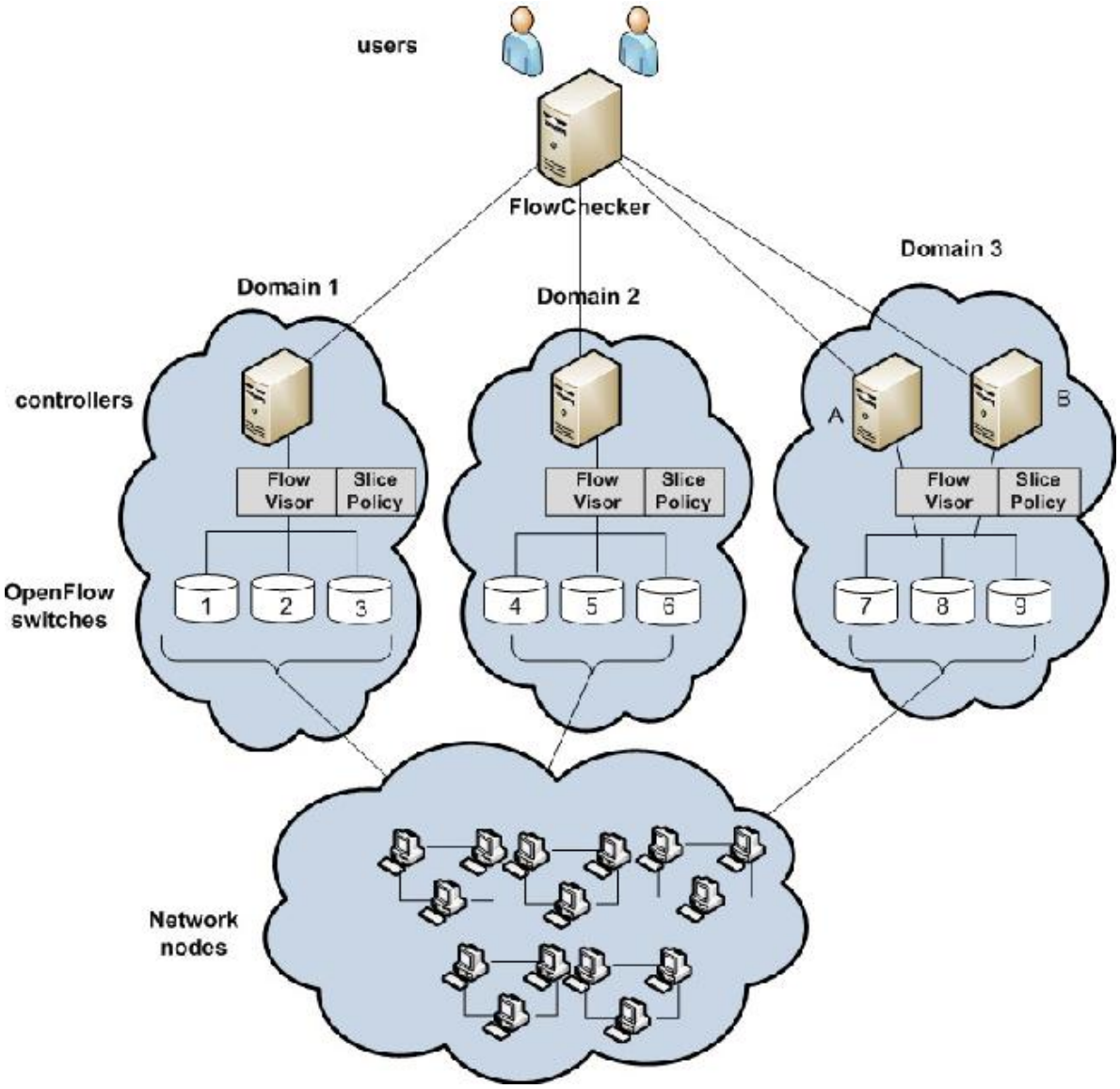


Рисунок 1.9 - Організація мережі SDN з використанням FlowVisor і FlowChecker

Додатки контролера SDN в основному розгортаються у масштабних сценаріях, що вимагає всебічної перевірки можливих помилок програмування. Система, що робить це, називається NICE, була описана в 2012 році. Запровадження універсальної архітектури безпеки вимагає комплексного та тривалого підходу до SDN. З самого початку розробники дивляться на можливі способи захисту мережі SDN, які не ставлять під загрозу масштабованість.

Доставка групових даних за допомогою SDN

Розподілені програми, що працюють в центрах обробки даних, зазвичай реплікують дані з метою синхронізації, відмовостійкості, балансування навантаження та наближення даних до користувачів (що зменшує затримку користувачів і збільшує їх пропускну спроможність). Крім того, багато програм, таких як Hadoop, повторюють дані в центрі обробки даних, щоб підвищити відмовостійкість та полегшити процес відновлення даних. Всі ці операції вимагають доставки даних з однієї машини або центру обробки даних до декількох машин або центрів обробки даних. Процес надійного доставки даних від одного комп'ютера до декількох машин називається RGDD.

Перемикачі SDN можуть бути використані для RGDD шляхом встановлення правил, що дозволяють пересилати до декількох вихідних портів. Наприклад, OpenFlow надає підтримку для групових таблиць починаючи з версії 1.1, що робить це можливим. Використовуючи SDN, центральний контролер може ретельно і розумно налаштувати пересилання дерев для RGDD. Такі логічні дерева можуть бути побудовані з урахуванням стану мережевої завантаженості для підвищення продуктивності. Наприклад, MCTCP є схемою для доставки даних до декількох вузлів в центрах обробки даних, які спираються на регулярні та структуровані топології, в той час як DCCast і QuickCast є подібними підходоми для швидкого і надійчного постачання даних через центри обробки даних.

Рівень даних SDN

Рівень даних включає в себе ресурси, які безпосередньо стосуються трафіку клієнтів, а також допоміжні ресурси для забезпечення належної віртуалізації, підключення, безпеки, доступності та якості. Блок ресурсів включає джерела даних, канали передачі даних та ядра обробки трафіку, а також віртуалізатор, чия функція полягає в абстракції ресурсів контролера SDN та виконуванні політики конфіденційності. Іноді також представляє

базову базу даних ресурсів (RDB) - концептуальне сховище всієї ресурсної інформації, відомої мережевому елементу [2]. Розмежування рівня даних і рівня управління зображено на рисунку 1.10.

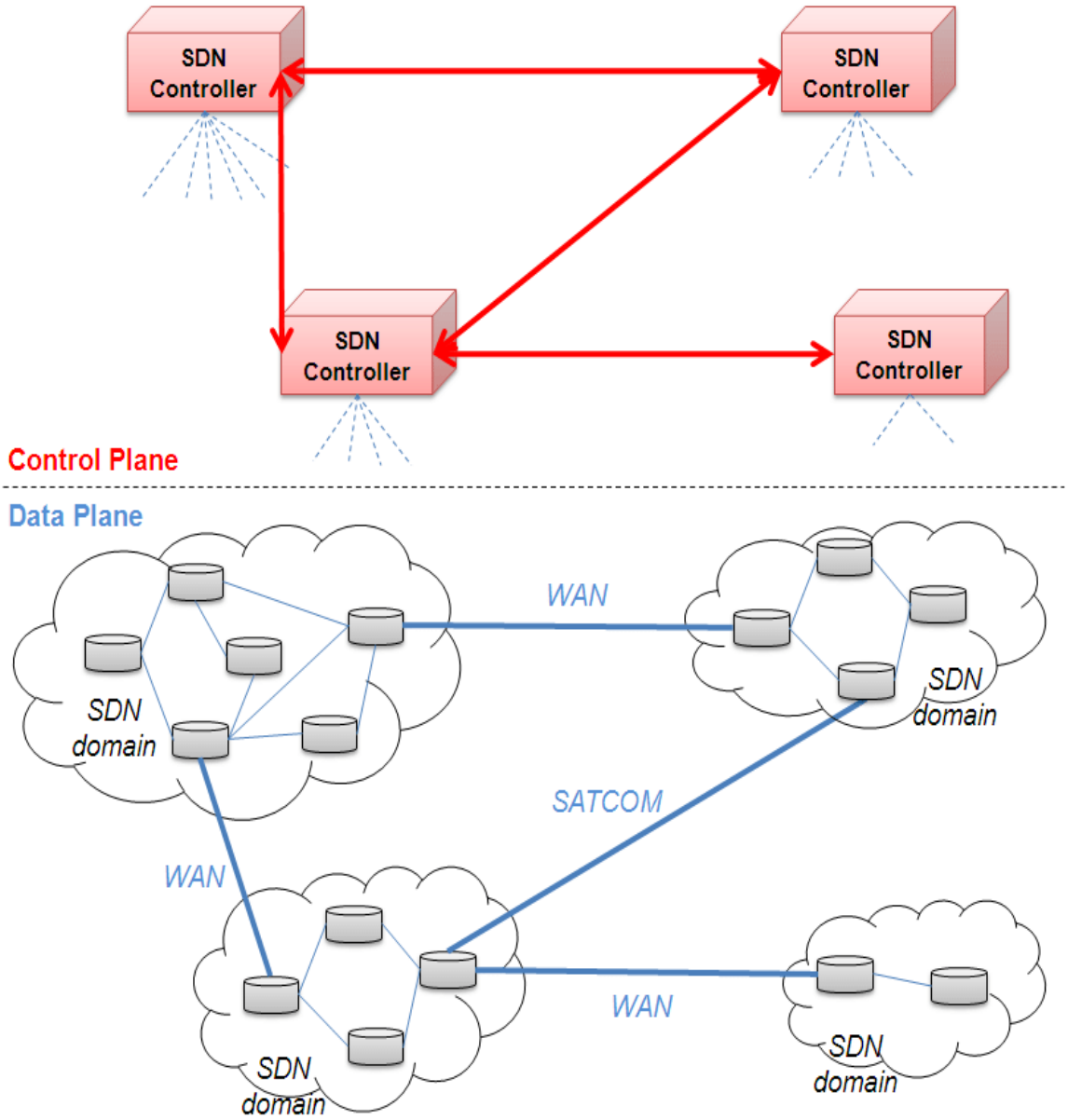


Рисунок 1.10 - Схематичне порівняння рівня управління і рівня даних

Програмне забезпечення, що визначається мережею, виконує функції, що пов'язані з передачею даних та обробкою трафіку, таких як QoS, фільтрація, моніторинг або натискання. Трафік може віходити або виходити з рівня даних SDN через фізичні або логічні порти, і може бути спрямований до або з функцій пересилання та обробки. Обробка трафіку може бути

пояснена на прикладі ядра OAM, функції шифрування або функції віртуалізованої мережі. Контроль за переадресацією або обробкою трафіку може виконуватися контролером SDN або окремими механізмами, які можуть бути організовані разом з даним контролером SDN. Схематичне порівняння рівнів управління і даних можна побачити на рисунку 1.9.

Рівень даних реалізує рішення по пересиланню трафіка, прийняті на рівні управління. Як правило, рівень даних не робить автономних рішень щодо організації потоків трафіку у мережі. Однак рівень управління може налаштувати рівень даних таким чином, щоб він автономно реагував на такі події, як мережеві збої або підтримку функцій доставки, наприклад, через LLDP, STP, BFD або ICMP [4].

Інтерфейс між рівнями даних та управління (D-CPI) включає такі функції, як

- Програмний контроль усіх функцій, що виявляються RDB
- Інформація про потенціальні можливості мережі
- Повідомлення про події

Агент рівня даних - це сутність, яка виконує інструкції контролера SDN на рівні даних. Координатор площини даних - це сутність, яка розподіляє ресурси даних для різних агентів клієнта та встановлює політики для керування ними. Агенти та координатори створені для досягнення однієї мети на кожному рівні архітектури.

На найнижчому рівні рекурсії ресурси рівня даних є фізичними (включаючи, наприклад, комутатори). Проте, на більш високих рівнях абстракції ресурси ресурсу даних не повинні бути фізичними. Як і на інших рівнях, архітектура SDN працює за абстрактною моделлю рівня даних. І якщо функції цієї моделі, правильно виконані, архітектура не має жодних відмінностей з цією моделлю.

Адміністратор вибирає, якими ресурсами повинен управляти даний контролер SDN. Ці ресурси представлені у вигляді набору віртуальних

серверів (VNEs), пов'язаних між собою для формування підмереж. VNE може бути послідовною абстракцією як підмережа.

Архітектура не накладає жодних обмежень на технології, що використовує рівень даних. Контролер SDN може бути використаний для програмування рівня даних, що реалізований як у вже існуючих технологіях, таких як DWDM, OTN, Ethernet, IP тощо, так і в більш нових технологіях рівня даних, які ще розвиваються.

OpenFlow

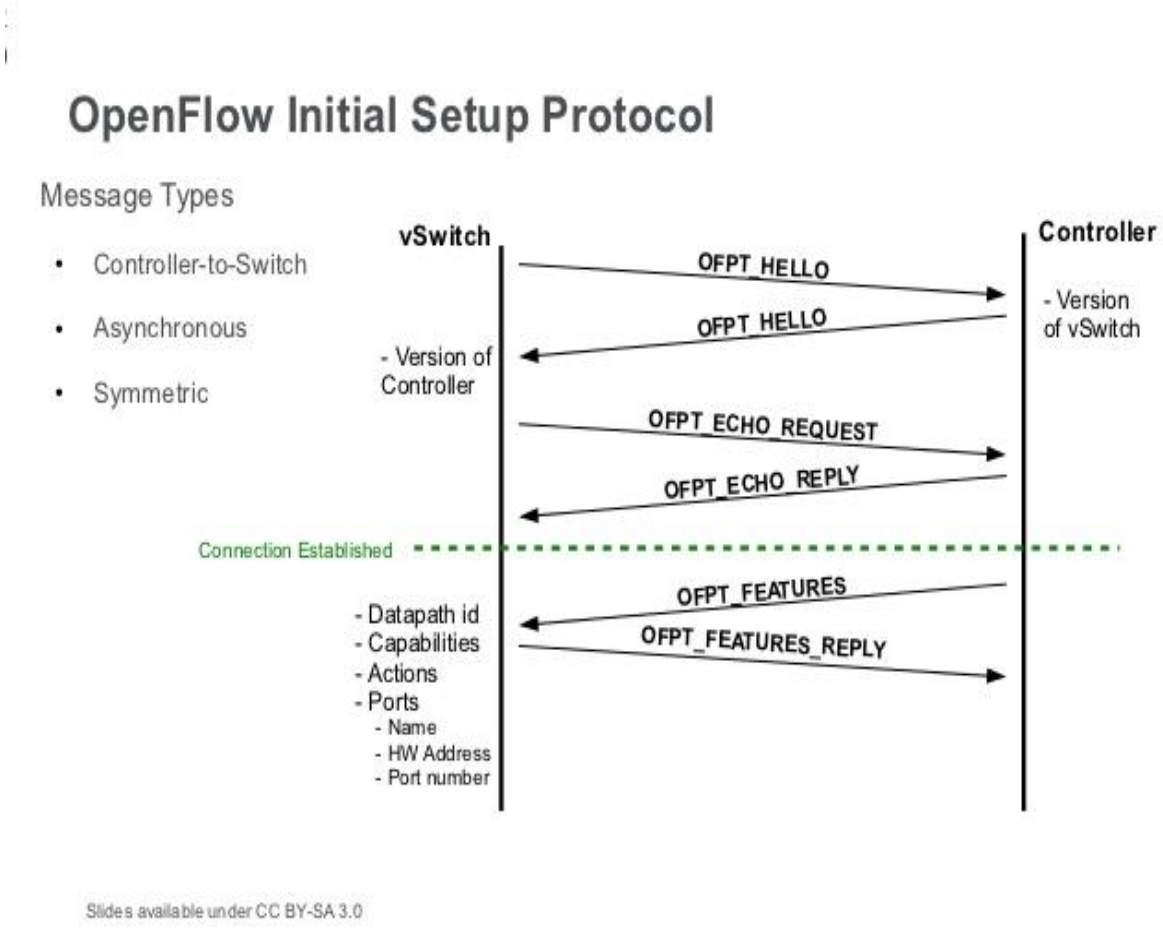


Рисунок 1.11 - Схема ініціації OpenFlow

OpenFlow (OF) вважається одним із перших стандартві SDN. Впочатку він визначив протокол зв'язку в середовищах SDN, який дає змогу рівню контроля SDN взаємодіяти з рівнем передачі даних, тобто з мережевими

пристроями, такими як комутатори та маршрутизатори, як фізичними, так і віртуальними, щоб вони могли краще адаптуватися до змін вимог.

Контролер SDN в SDN - це "мозок" мережі SDN, що служить для передачі інформації між пристроями низьких рівнів (комутаторами/маршрутизаторами) та додатків та логічної складової, тобто високим рівнем. На рисунку 1.11 можна побачити схему ініціації зв'язку між комутатором і рівнем управління. Останнім часом, оскільки організації розгортають більше мереж SDN, для контролерів SDN було поставлено завдання об'єднати між доменами контролера SDN, використовуючи загальні інтерфейси додатків, такі як OpenFlow та відкрити базу даних віртуальних комутаторів (OVSDB).

Для роботи в середовищі OF будь-який пристрій, який хоче встановити зв'язок з контролером SDN, має підтримувати протокол OpenFlow. Завдяки цьому інтерфейсу контролер SDN записує зміни до таблиці потоку комутаторів/маршрутизаторів, що дозволяє мережевим адміністраторам розподіляти трафік, керувати потоками для оптимальної продуктивності та починати тестування нових конфігурацій та програм.

OpenFlow дозволяє мережевим контролерам визначати шлях пакетів через мережу комутаторів [4]. Контролери відрізняються від комутаторів. Таке відокремлення управління від передачі даних дозволяє більш складне управління трафіком, ніж це можливо, використовуючи списки контролю доступу (ACL) та протоколи маршрутизації. Крім того, OpenFlow дозволяє використовувати комутатори від різних постачальників - часто з власними інтерфейсами та мовами програмування - для керування віддалено за допомогою єдиного, відкритого протоколу. Винахідники протоколу вважають OpenFlow головним протоколом розвитку SDN.

OpenFlow дозволяє віддалене адміністрування таблиць передачі пакетів комутаторів третього рівня, додавання, редагування та видалення правил та дій, що пов'язані з передачею пакетів. Таким чином, рішення маршрутизації можуть бути здійснені контролером з використанням заздалегідь

встановлених правил або прийнятих динамічно. Пакети, які не відповідають комутатору, можна передати контролеру. Контролер може потім вирішити змінити існуючі правила таблиці потоків для одного або декількох комутаторів або розгорнути нові правила, щоб запобігти структурному потоку трафіку між комутатором та контролером.

Протокол OpenFlow розташований на вищих рівнях порівняно з TCP і передбачає використання TLS. Контролери повинні слухати TCP-порт 6653 для комутаторів, які хочуть встановити з'єднання. Ранні версії протоколу OpenFlow неофіційно використовували порт 6633.

Переваги OpenFlow:

1) можливість керувати мережею програмно, дозволяє прискорено впроваджувати нові функції та послуги;

2) абстракція; відокремлення обладнання та програмного забезпечення, рівня управління та пересилання даних, а також фізична і логічна конфігурація.

3) централізований інтелект, спрощує підготовку мережі, оптимізує продуктивність, деталізує політику управління.

Висновки до розділу 1

Проведено аналіз основних принципів побудови SDN. Проаналізовано передумови виникнення SDN, з яких можна зробити висновок, що концепція програмно-конфігуровних мереж має великий потенціал і допомагає вирішити проблеми, що зустрічаються в класичних підходах до організації мереж, зокрема зменшити витрати на організацію мережі за рахунок використання більш простих пристроїв і спростити контроль за мережею, за рахунок чіткого відокремлення рівня даних і рівня управління.

В розділі проаналізовані особливості загальної архітектури мережі, що побудована за даною концепцією і ключові деталі.

Досліджені основні протоколи для організації цього виду мереж, а також основні типи додатків.

2. АНАЛІЗ КОНФІГУРАЦІЙ ТА ОСОБЛИВОСТЕЙ ФУНКЦІОНУВАННЯ РІВНЯ УПРАВЛІННЯ SDN

2.1 Архітектура рівня управління

Рівень управління SDN складається з одного або декількох контролерів (вузлів) SDN. Не дивлячись на те, що деякі функції управління можуть виконувати модулі на інших рівнях, більшість функції контролера неможливо перенести на блоки з інших частин архітектури.

Архітектура не визначає внутрішній дизайн контролера SDN.

- Контролер зазвичай контролює область мережі (підмережу), що охоплює більше одного фізичного вузла.
- Контролер не має ніякого зв'язку з іншими вузлами і пристроями мережі. Контролер може управляти тільки вузлами з підмережі, що була виділена для нього.

Функції та служби, що входять до зовнішньої поведінки контролера, включають в себе повну видимість інформаційної моделі. Додаткові функції, які можуть бути необхідні, залежно від обставин:

- знання топології та обчислення шляху (контролер може також викликати зовнішній сервіс для цих функцій);
- створення та підтримка абстрактної моделі ресурсів для своїх додатків, з ресурсами, що обмежені примусовою політикою.

Контролер SDN буде координує ряд взаємопов'язаних ресурсів, які часто поширюються через кілька платформ, а іноді і для забезпечення транзакційної цілісності як частини процесу.

Архітектура SDN не визначає внутрішню структуру або реалізацію контролера SDN. Це може бути:

- один процес;
- група процесів, що розділяють між собою навантаження або замінюють один одного у випадку переривання одного з процесів;
- сукупність окремих функціональних компонентів у спільній організації;

- контролер може співпрацювати з зовнішніми службами для деяких своїх функцій, наприклад, для обчислення шляху.

Допускається будь-яка комбінація цих альтернатив: контролер SDN розглядається без втручання в деталі реалізації, а визначений тільки зовнішньою поведінкою. Компоненти контролера можуть працювати на довільних обчислювальних платформах, включаючи виділення локальних ресурсів для фізичного вузла. Вони також можуть виконуватись на розподілених та, можливо, тимчасових ресурсах, таких як віртуальні машини (VM) в центрах обробки даних.

Архітектура описує зовнішню поведінку контролера SDN. Одним з основних принципів роботи контролера є організація взаємодії елементів з підмережі з глобальною мережею. Окрім того, контролери SDN повинні організовувати роботу між компонентами підмережі і зовнішніми базами даних. Кілька диспетчерських або контролерних компонентів можуть мати спільний доступ до запису до мережевих ресурсів, але щоб відповідати принципам SDN, вони повинні:

a) бути налаштовані на керування розподіленими наборами ресурсів або дій;
b) бути синхронізованими один з одним, щоб уникати несумісні або конфліктуючі команди.

При розподіленому керуванні розподіленими мережевими ресурсами, жорстка синхронізація станів призводить до надмірної стійкості ата складності.

Припущення про внутрішню узгодженість в контролері відрізняється від питання про узгодженість станів з точки зору базових ресурсів даних. Завжди очікується, що контролери SDN зможуть працювати з подібними подіями, які доступні з різних частин інфраструктури.

Не маючи наміру мінімізувати їх значення, такі проблеми, як завантаження, синхронізація, міграція, резервне копіювання, аудит, керування випуском програмного забезпечення контролера тощо, є

внутрішніми для контролера SDN чорного ящика, і це не стосується архітектури SDN.

Не дивлячись на те, що контролер SDN позначений "чорним ящиком", необхідно визначити мінімальний набір функціональних компонентів в контролері SDN, а саме функцію керування рівнем (DPCF), координатором, віртуалізатором та агентом. Залежно від вимоги логічної централізації, контролер SDN може включати довільні додаткові функції. База даних ресурсів (RDB) моделює поточну інформаційну модель та необхідні допоміжні можливості.

Функція управління рівнем даних

Компонент DPCF ефективно управляє виділеними для нього ресурсами та використовує їх відповідно до інструкцій координатора або віртуалізаторів, які їх контролюють. Ці ресурси набувають форму інформаційної моделі, доступ до якої здійснюється через агента на підпорядкованому рівні. Оскільки обсяг регулятора SDN повинен охоплювати декілька (віртуальних) вузлів або навіть декілька віртуальних мереж (з окремим екземпляром D-CPI для кожного), DPCF має включати функцію, яка оперує об'єднаннями ресурсів. Ця функція зазвичай називається управляючою. Архітектура не визначає її як окремий функціональний компонент. На рисунку 2.1 зображена схема роботи DPCF.

Координатор

Для налаштування як клієнтських, так і серверних середовищ потрібна функціональність керування. Координатор є функціональним компонентом контролера SDN, який виконує ці функції. Клієнти та сервери вимагають управління всіма видами даних, на усіх рівнях, тому функціональні блоки координаторів є повсюди. Концепція віртуалізації мережевої функції відрізняється від концепції віртуалізації, яка використовується в архітектурі SDN. В архітектурі SDN віртуалізація - це розподіл абстрактних ресурсів конкретним клієнтам або додаткам; в NFV метою є абстрагування мережевих

функцій від виділеного обладнання, наприклад, щоб вони могли бути розміщені на серверних платформах у хмарних центрах даних.

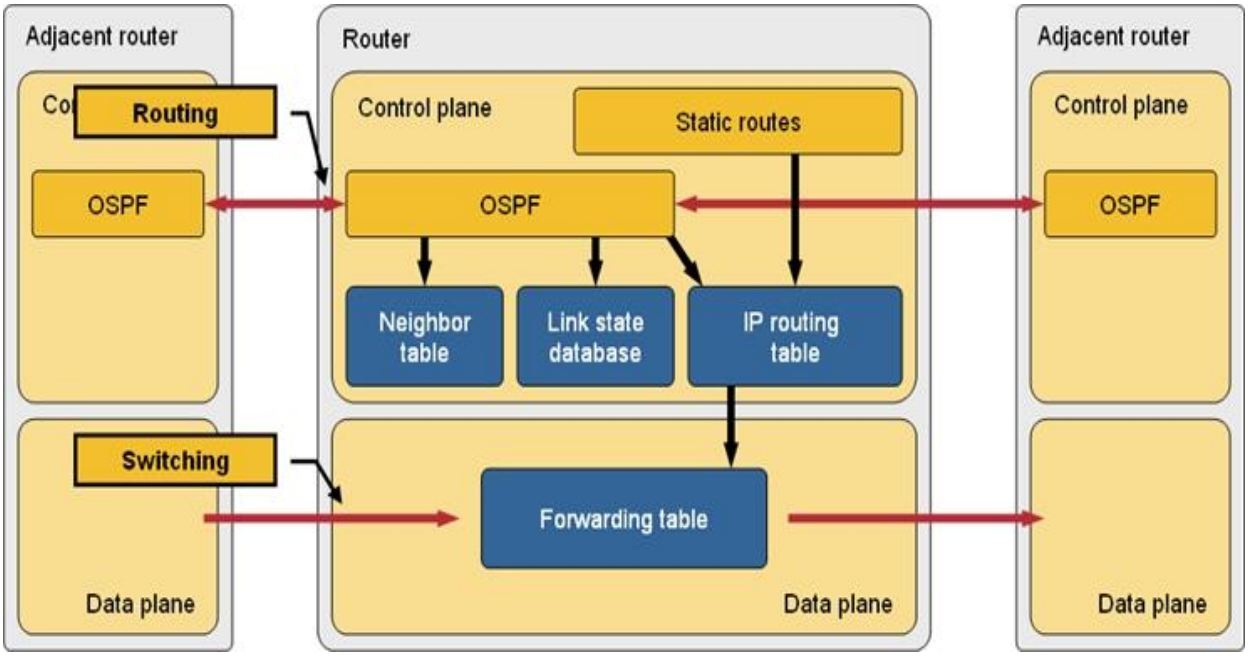


Рисунок 2.1 - Робота функції управління рівнем даних

Віртуалізатор

Контролер SDN пропонує послуги додаткам як представлення інформаційної моделі, що об'єднує основні ресурси, політики, що встановлюється, та локальні або зовнішні функції підтримки. Функціональний об'єкт, який підтримує це представлення і опис інформаційної моделі на A-CPI (інтерфейс рівня контролера), називається віртуалізатором. Він представляє локальний домен для забезпечення видимості мережі клієнтом. Віртуалізатор створюється координатором для кожної клієнтської програми або організації. Координатор виділяє ресурси, які використовує віртуалізатор для представлення A-CPI, який він видає клієнтам додатку, і він встановлює політику, яку застосовує віртуалізатор. Ефект від цих операцій полягає у створенні агента для даного клієнта. Віртуалізатор може розглядатися як процес, який отримує запити клієнта через A-CPI, перевіряє запити щодо політики та ресурсів, призначених

клієнту, перекладає запит на умови основних ресурсів і передає результати до DPCF та D-CPI. Віртуалізатор, DPCF та інші функції контролера SDN співпрацюють, щоб забезпечити такі функції, як інтерпретація запитів, спільне використання ресурсів, служби неявного постачальника та цілісність транзакцій. На рисунку 2.2 можна побачити приклад типової архітектури NVF. Також, на рисунку 2.3 схематично представлена фабрика SDN з використанням NVF.

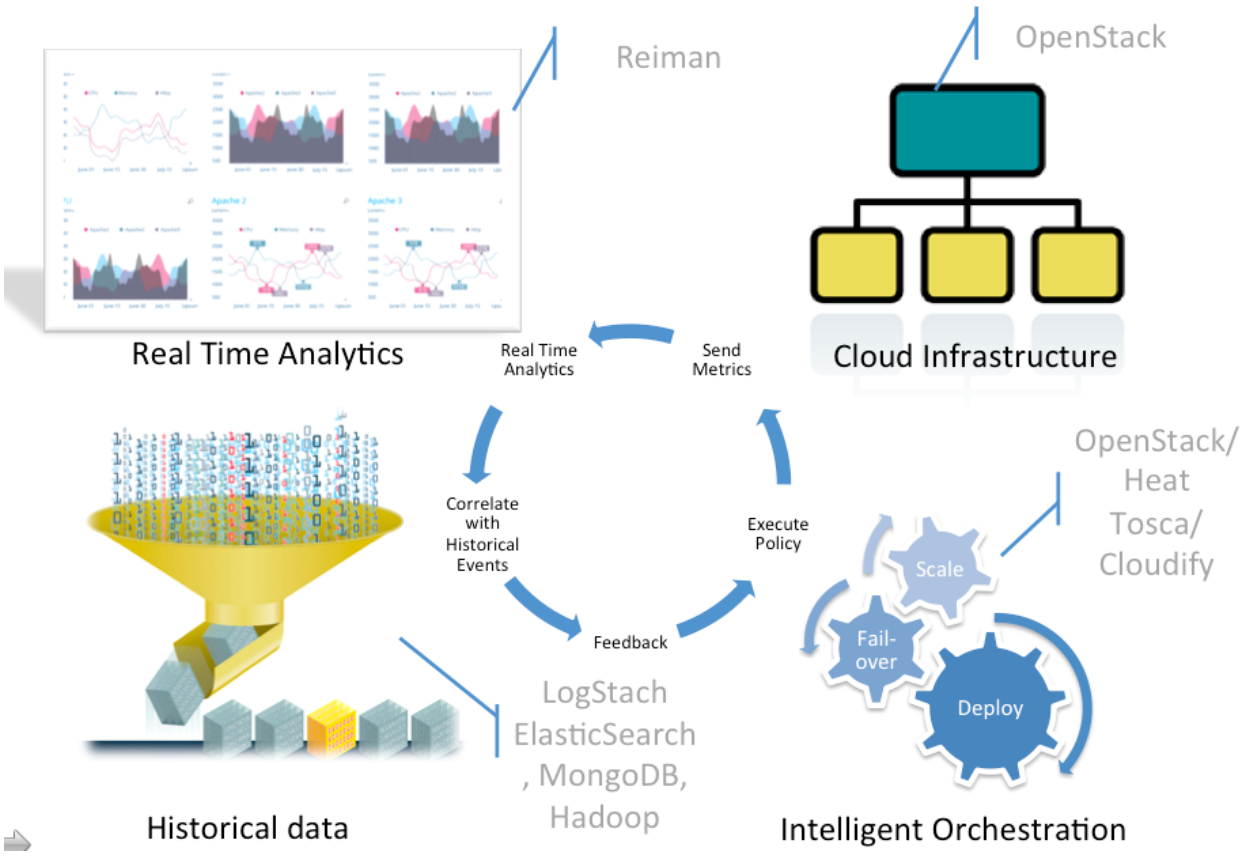


Рисунок 2.2 - Архітектура NFV

Агент

Модель контролера-агента підходить для взаємозв'язку між контрольованою та контролюючою сутностями і застосовується рекурсивно до архітектури SDN. Контрольований об'єкт визначається агентом, функціональним компонентом, який представляє ресурси та можливості

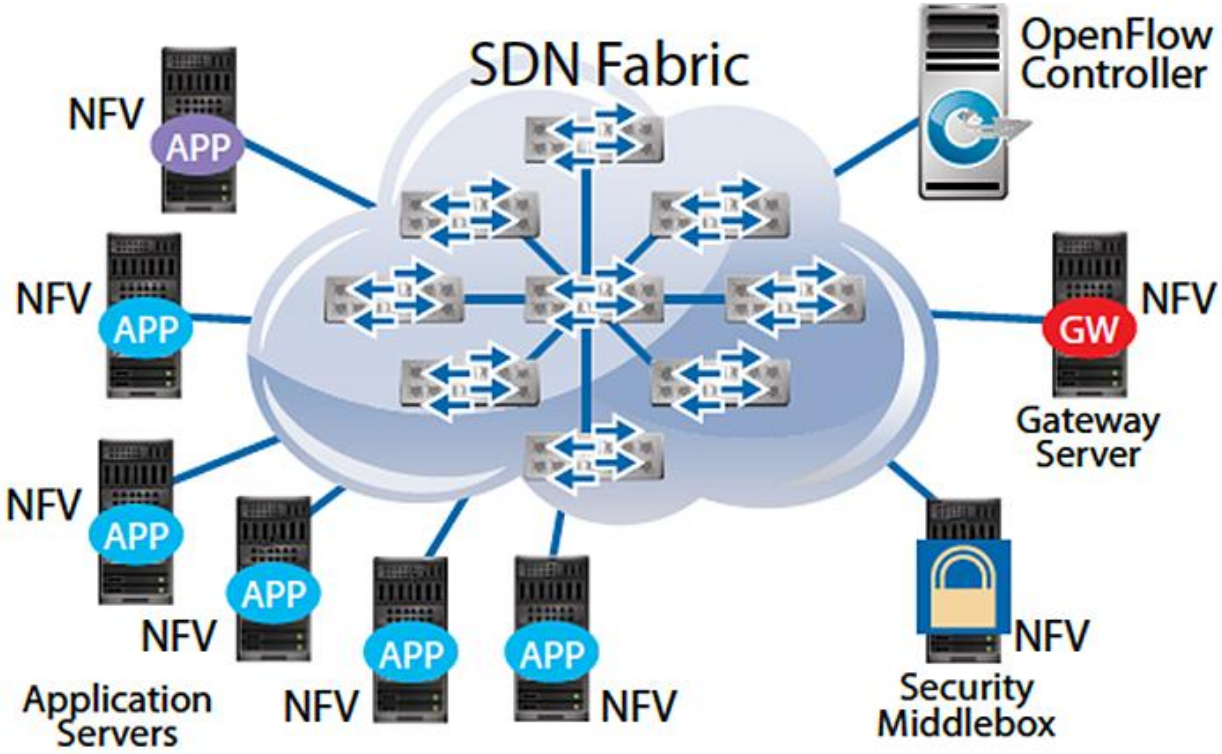


Рисунок 2.3 - SDN фабрика з використанням NFV

клієнта в середовищі сервера. Агент у даному контролері SDN на рівні N представляє ресурси, що доступні для клієнта або застосування контролера SDN, на рівні N+1. Агент на рівні даних N-1 представляє ресурси та дії, доступні для даного контролера рівня SDN. Навіть незважаючи на те, що фізичне місцезнаходження агента розташовується всередині довіреного домену сервера (тобто на платформі контролера сервера SDN), агент асоційовано перебуває в довіреному домені клієнта.

Інші компоненти контролера

Для уникнення великого списоку вимог, архітектура описує лише функції, необхідні для контролера SDN, але не виключає додаткових функцій. Вони можуть мати форму додатків або функцій, що підтримуються контролером. Ці функції можуть експортуватися вибірково до деяких або одночасно до всіх клієнтів зовнішніх додатків сервера, або використовуватись внутрішньо адміністратором провайдера для власних цілей.

Як компоненти контролера SDN, так і програми або функції підлягають синхронізації, як й інші компоненти контролера. Щоб полегшити інтеграцію зі стороннім програмним забезпеченням, інтерфейси до таких програм або функцій можуть бути такими самими, як і інші у A-CPI.

Особливості безпеки вбудованих додатків повинні бути відомі рівню управління. Оскільки вони виконуються в довіреному домені сервера, вони проходять ряд перевірок.

Делегування управління

Незважаючи на те, що ключовий принцип SDN вказується як розмежування рівня управління з рівнем даних, зрозуміло, що агент на рівні даних самостійно контролює процеси від імені контролера SDN. Крім того, ряд функцій з аспектами управління як правило виконуються на елементах мережі, наприклад, OAM, обробка ICMP, розпізнавання дефектів та інтеграція.

Використання принципу роз'єднання дає змогу контролеру SDN делегувати функції керування рівню даних за умови, що ці функції ведуть себе у спосіб, прийнятний для контролера. Тобто контролер дозволяє агенту виконувати певний список функції. Такий принцип є життєво важливим способом застосування принципів SDN до реального світу. Критерії, які заохочують контролер до делегування функції рівню даних, включають:

- швидкодію в режимі реального часу, необхідну для мережевих подій;
- великий обсяг трафіку, який потрібно обробляти;
- байтово- або бітово-орієнтовані функції, які важко поділити на пакети;
- передбачувана, добре зрозуміла, повністю стандартизована поведінка, що часто повторюється і небагато коштую. Наприклад, шифрування, вставка BIP, AIS, навчання MAC, обмін CCM;
- стійкість до збоїв або безперервність у випадку несправності контролера чи повторної ініціалізації;
- необхідність використовувати функціональність, що доступна тільки

на рівні даних;

- відсутність можливості відокремлення функції.

Припускаючи, що вихідні дані можуть бути доступними, контролер SDN завжди має можливість не делегувати контрольну функцію, а сам проводити необхідні операції. Перераховані вище критерії впливають на те, чи є такий вибір практичним. Вирішуючи, чи делегувати функцію, контролер SDN повинен зрозуміти, чи поведінка делегованої функції кандидата повністю задовольняє її потреби. Це може вимагати вбудованих або налаштованих знань та/або запитів щодо функціональних можливостей та можливостей кандидата.

Як правило, контролер отримує інформацію про стан виконання делегованих функції. Делегована функція управління може повідомляти про:

- зміни стану та значення атрибута;
- перевищення допустимих значень навантаження;
- помилки обладнання, відновлення або встановлення, оскільки така діяльність впливає на ресурси, що знаходяться в компетенції контролера SDN.

Делегована функція управління на рівні даних може виконувати стандартизовані протоколи та повідомляти про проміжні або кінцеві результати, винятки або зміни стану. Приклади включають:

- шифрування трафіку, включаючи обмін ключами та оновлення;
- CFM: 802.1ag або BFD;
- агент аутентифікації 802.1X;
- GMPLS, що надає контроллеру SDN доступ до сигналів через межі адміністративних доменів, де SDN може підтримуватися або не підтримуватися.

Коли контролери спілкуються в межах адміністративного домену, безпека та питання довіри та приховування інформації виходять на перший план. Інформація, яку контролери передають один одному, може включати в себе наступне:

- інформація про сусідні контролери;
- топологія мережі, в межах дозволених політиками конфіденційності;
- інформація про стан та атрибути, включаючи можливість підписки на сповіщення про зміну стану та атрибуту, за узгодженням з політикою конфіденційності;
- інформація про пересилання, наприклад, доступність даних на тому чи іншому рівні;
- інформація про розрахунок шляху, наприклад, вартість маршруту, політика захисту та відновлення;
- інша інформація, така як конфігурація OAM, оцінка QoS та звітність, інформація про виставлені рахунки.

Операції потрібно синхронізувати одну з одною. Ці інформаційні обміни, як правило, сумісні з мережевими доменами, що не є SDN, які використовують існуючі протоколи.

Існуючі рішення SDN

OpenDaylight

Проект OpenDaylight є спільним проектом з відкритим кодом, розміщеним у фонді The Linux Foundation. Метою проекту є сприяння розробці програмного забезпечення (SDN) та віртуалізації мережевих функцій (NFV). Програмне забезпечення написано на мові програмування Java.

OpenDaylight підтримує технологію OpenFlow. Перший код проекту OpenDaylight під назвою Hydrogen був випущений в лютому 2014 року.

Репозиторій вихідного коду включає в себе вбудований вихідний код, який спочатку створювався Big Switch Networks, Cisco та NEC. В даний час існує понад 1000 сукупних внесків від різних організаційта осіб. Існує багато документації пов'язаної з роботою OpenDaylight. Ці ресурси спрямовані на розробників, які бажають внести свій внесок у проект, а також інших, котрі зацікавлені у вивченні конкретних підпроектів.

Контролер OpenDaylight може розгортатись у різних середовищах мережі. Він може підтримувати структуру модульного контролера, але може забезпечити підтримку інших стандартів SDN та майбутніх протоколів.

Також контролер OpenDaylight розширює відкриті північні API, які використовуються додатками. Ці програми використовують контролер для збору інформації про мережу, запуск алгоритмів для проведення аналітики, а потім за допомогою контролера OpenDaylight створювати нові правила у всій мережі.

Контролер OpenDaylight реалізований виключно в програмному забезпеченні і зберігається у власній віртуальній машині Java (JVM). Це означає, що він може бути розгорнений на апаратній та операційній системних платформах, які підтримують Java. Місце OpenDaylight в структурі SDN представлено на рисунку 2.3.

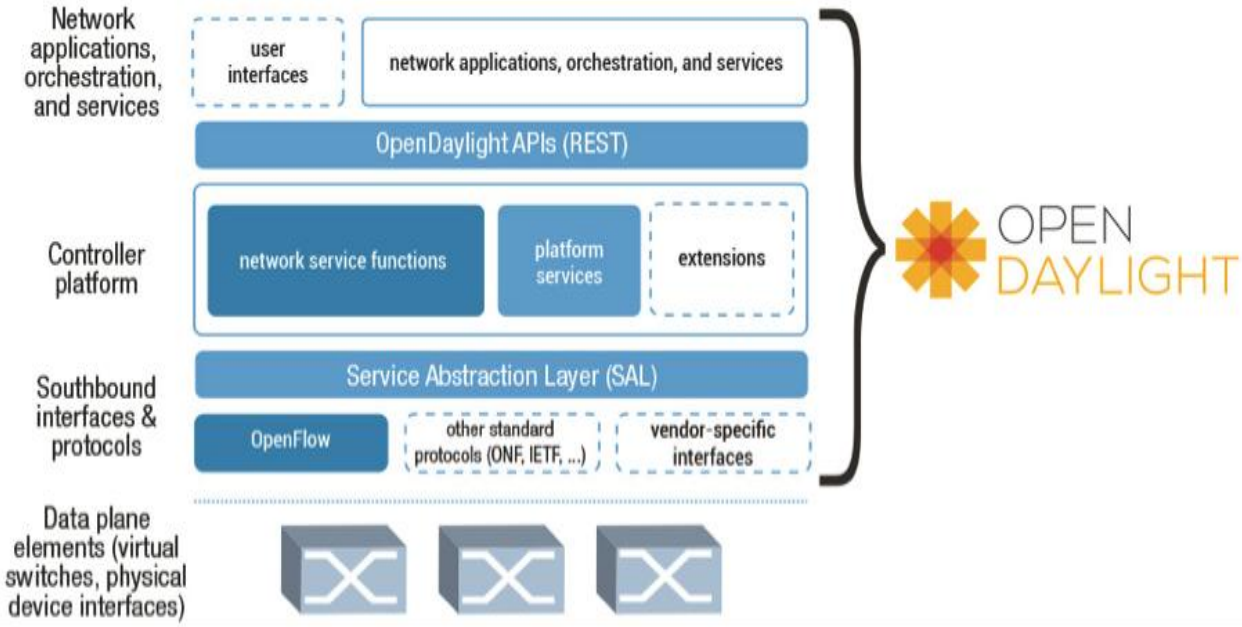


Рисунок 2.3 - структура OpenDaylight

Open vSwitch

Open vSwitch, іноді скорочений як OVS, - це реалізація з відкритим кодом розподіленого віртуального багаторівневого комутатора. Основна мета Open vSwitch полягає в тому, щоб надати комутаційний стек для середовищ

віртуалізації обладнання, одночасно підтримуючи кілька протоколів та стандартів, що використовуються в комп'ютерних мережах. На рисунку 2.4 представлена архітектура OpenSwitch.

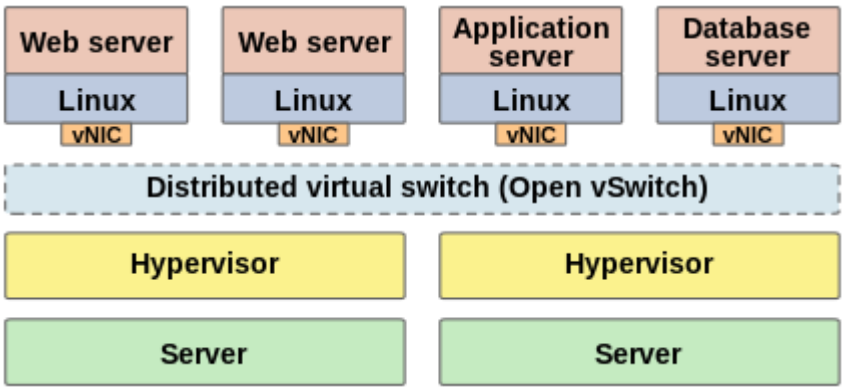


Рисунок 2.4 - Архітектура OpenSwitch

Це програмна реалізація віртуального багатошарового мережевого комутатора, призначеного для ефективної автоматизації мережі через програмні розширення, одночасно підтримуючи стандартні інтерфейси управління та протоколи, такі як NetFlow, sFlow, SPAN, RSPAN, CLI, LACP і 802.1ag. Крім того, Open vSwitch призначений для забезпечення прозорого розподілу на кількох фізичних серверах, дозволяючи створювати перехресні сервери таким чином, щоб абстрагуватися від базової архітектури сервера, подібно до VMware vNetwork, що розповсюджується разом із переходом або Cisco Nexus 1000V.

Open vSwitch може функціонувати як програмний мережевий комутатор, який працює в рамках гіпервізора віртуальної машини (VM), а також, як стек керування для спеціалізованого комутаційного обладнання. В результаті, він був перенесений на кілька платформ віртуалізації, комутацію чіпсетів та апаратні прискорювачі мережі. Відкритий vSwitch - це мережевий комутатор за замовчуванням на платформі віртуалізації XenServer з моменту його версії 6.0, а також у Xen Cloud платформі за допомогою інструментальної панелі керування XAPI. Він також підтримує Xen, Linux KVM, Proxmox VE і VirtualBox гіпервізорів, а порт для Hyper-V також

доступний. Відкритий vSwitch також інтегрований в різні платформи для платформ хмарних обчислень та системи керування віртуалізацією, включаючи OpenStack, openQRM, OpenNebula і ovitr.

Реалізація ядра Linux з Open vSwitch була об'єднана в основну частину ядра у версії 3.3 ядра, яка була випущена 18 березня 2012 року.

Більшість вихідного коду Open vSwitch написано на незалежній від платформи C мові, що забезпечує зручність переносу в різних середовищах.

OpenStack

OpenStack дозволяє користувачам розгортати віртуальні машини, які обробляють різні завдання для керування хмарним середовищем на льоту. Представлення архітектури OpenStack можна побачити на рисунку 2.5.

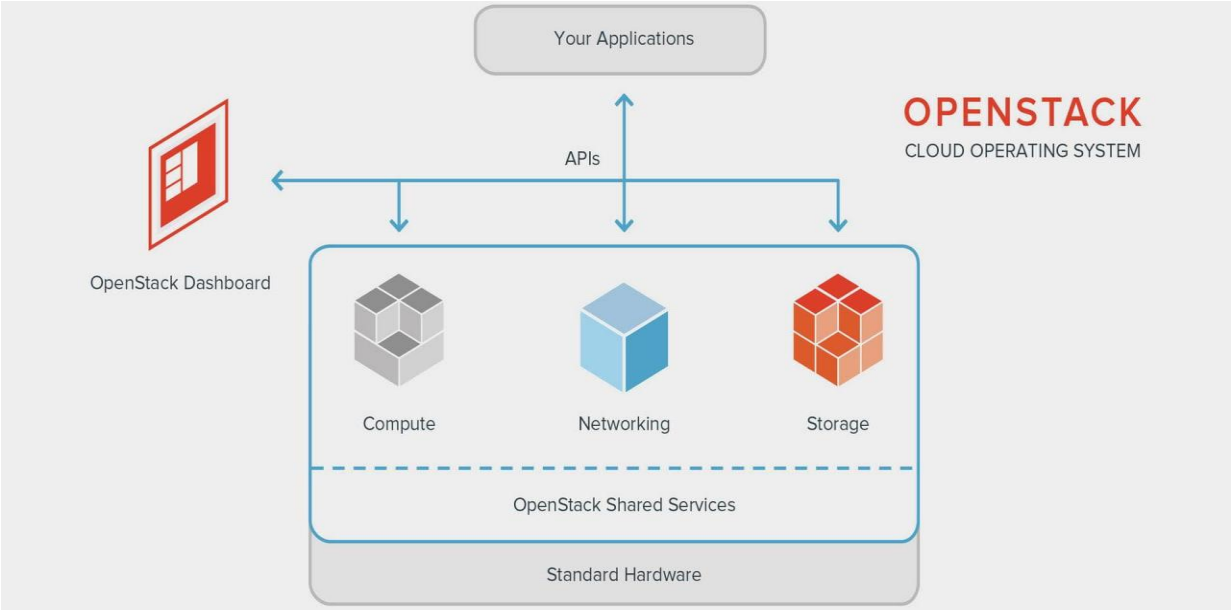


Рисунок 2.5 - Архітектура OpenStack

Це робить горизонтальне масштабування легким, а це означає, що завдання, які виграють від паралельного виконання, можуть легко обслуговувати більше або менше користувачів на льоту, просто створюючи більше віртуальних машин. Наприклад, мобільна програма, яка потребує зв'язку з віддаленим сервером, може розділити взаємодію з кожним

користувачем між різними екземплярами віртуальних машин. Таку систему можна швидко і легко масштабувати, не беручи до уваги кількість активних користувачів.

І найголовніше, OpenStack - це програмне забезпечення з відкритим кодом, а це означає, що кожен, хто використовує цю технологію, може отримати доступ до вихідного коду, внести будь-які зміни та модифікації, які вони потребують, і вільно надавати доступ до цих змін до спільноти в цілому.

Хмара використовується для забезпечення обчислень для кінцевих користувачів у віддаленому середовищі, де програмне забезпечення працює як служба на надійних та масштабованих серверах, а не на кожному комп'ютері кінцевого користувача. Хмарні обчислення бувають різних видів, проте останнім часом популярна концепція, коли та чи інша частина архітектури працює як служба. Наприклад, програмне забезпечення, платформа та інфраструктура. OpenStack потрапляє до останньої категорії і розглядається як інфраструктура як служба. OpenStack прискорює додавання нового екземпляра користувачами, з яким можуть працювати інші хмарні компоненти. Зазвичай інфраструктура запускає "платформу", на основі якої розробник може створювати програмні додатки, які доставляються кінцевим користувачам.

NVF

NFV (мережева функція віртуалізації) - головна ідея полягає в програмній імітації фізичних пристроїв, таких як маршрутизатори, комутатори тощо. Як і у випадку віртуалізації обчислень або зберігання, процес віртуалізації мережевих функцій однаковий. Кожна функція фізичного пристрою перетворюється на функцію програмного забезпечення, яка може працювати на будь-якому пристрою. Вибір, який саме пристрій яку програмну функцію буде виконувати, може змінюватися під час роботи без зміни програмного коду.

Сьогодні компанії покладаються на індивідуально налаштовану інфраструктуру, жодна інфраструктура не працює так само, як інфраструктура іншої компанії. В більшості випадків, подібний підхід є перевагою, оскільки дозволяє найкращим чином налаштувати внутрішні процеси. NFV не є стандартом або продуктом. Проте існують декілька організацій, що намагаються визначити стандарти для NFV, такі як ETSI. На даний момент існують тільки загальні рекомендації щодо організації NFV, що базуються на досвіді постачальників хмарних сервісів.

Досить часто NFV базується на даних компонентах.

1) Хмарна інфраструктура - OpenStack в даний час є найпопулярнішим вибором для цієї мети.

2) Функції SDN - Це суміш існуючих мережевих функцій, наданих у вигляді програмного забезпечення, а також абсолютно нових, що дозволяють легше керувати мережею.

3) Організаційне ядро - Відповідальне за забезпечення мережевих функцій у хмарній інфраструктурі.

4) Аналітичний движок - движки аналітики в основному є зворотним зв'язком. Це важливий елемент для визначення тогочи відповідають вимогам бажані SLA, а також як засіб для аналізу та оптимізації навантаження. У контексті NFV, де багато розуміння та рішень повинні відбуватися в режимі реального часу, движок аналітики є важливою функцією.

Різниця між SDN і NFV

Іноді важко помітити різницю між SDN і NFV, проте, не варто плутати ці поняття.

Як показано на рисунку 2.6, NFV та SDN доповнюють одне одного, проте вони не повинні бети залежними одна від одної. NFV може бути реалізована без використання SDN, хоча ці два поняття часто поєднують, і, для більшої ефективності необхідно використовувати обидва підходи.

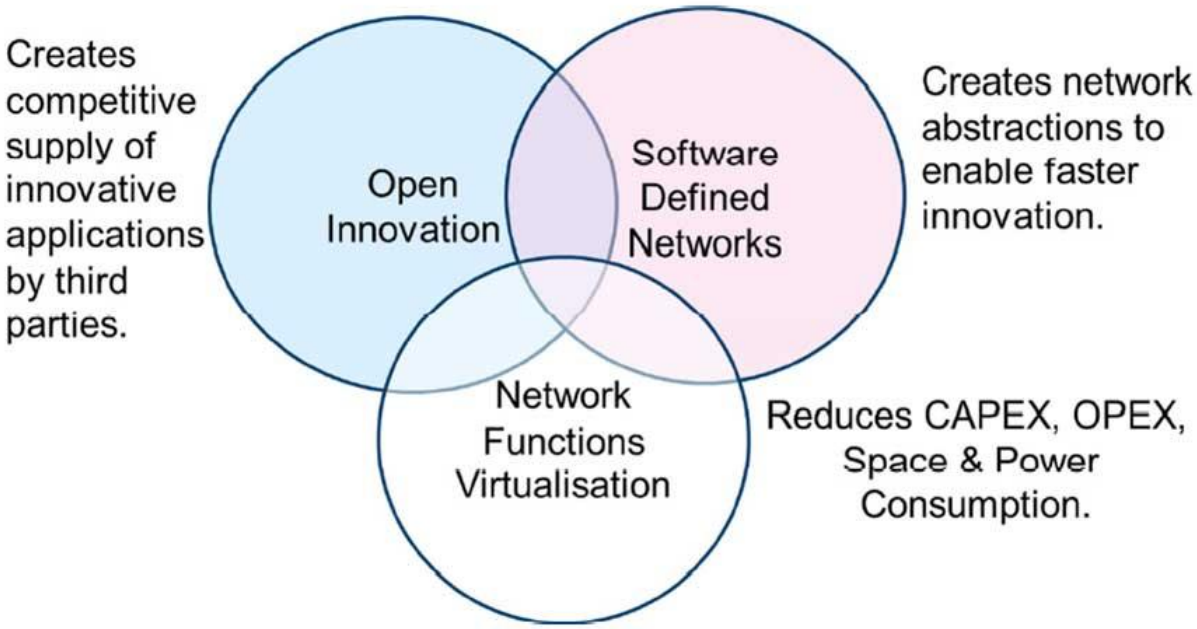


Рисунок 2.6 - Відношення NFV до SDN

Основні концепції NFV можуть бути реалізовані без використання механізмів SDN, спираючись на методи, що використовуються в даний час у багатьох центрах обробки даних. Але підходи, що спираються на відокремлення рівня управління та передачі даних, як це пропонується SDN, можуть підвищити продуктивність, спростити сумісність із існуючими системами та полегшити процедури експлуатації та технічного обслуговування. NFV може підтримувати SDN, забезпечуючи інфраструктуру, на якій можна запустити програмне забезпечення SDN. Крім того, NFV повністю відповідає вимогам SDN щодо використання віртуальних серверів і комутаторів.

Таким чином, можна зробити висновки що для досягнення сумісності і продуктивності необхідно використовувати обидва принципи. На рисунку 2.7 зображена традиційна реалізація управління маршрутизацією.

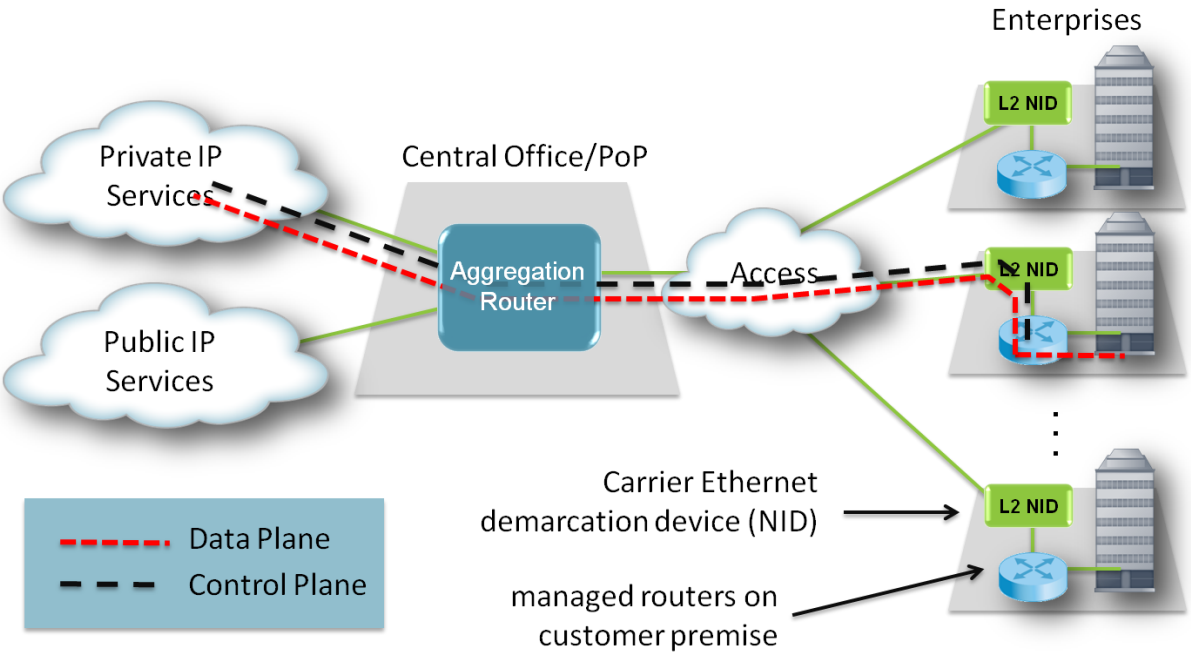


Рисунок 2.7 - Управління маршрутизацію за традиційним підходом

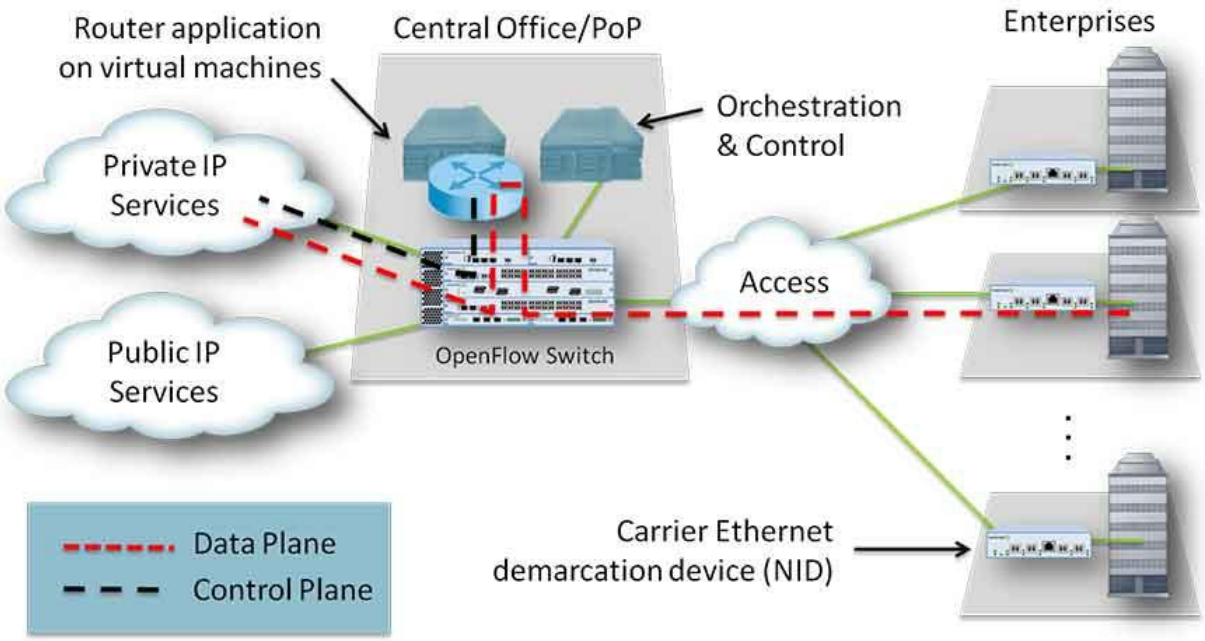


Рисунок 2.8 - Управління маршрутизацію з використання NFV

NFV можна застосувати шляхом віртуалізації функції маршрутизатора, як показано на рисунку 2.8. Все, що залишилося на стороні клієнта, - це мережевий пристрій для забезпечення розмежування, а також для вимірювання продуктивності.

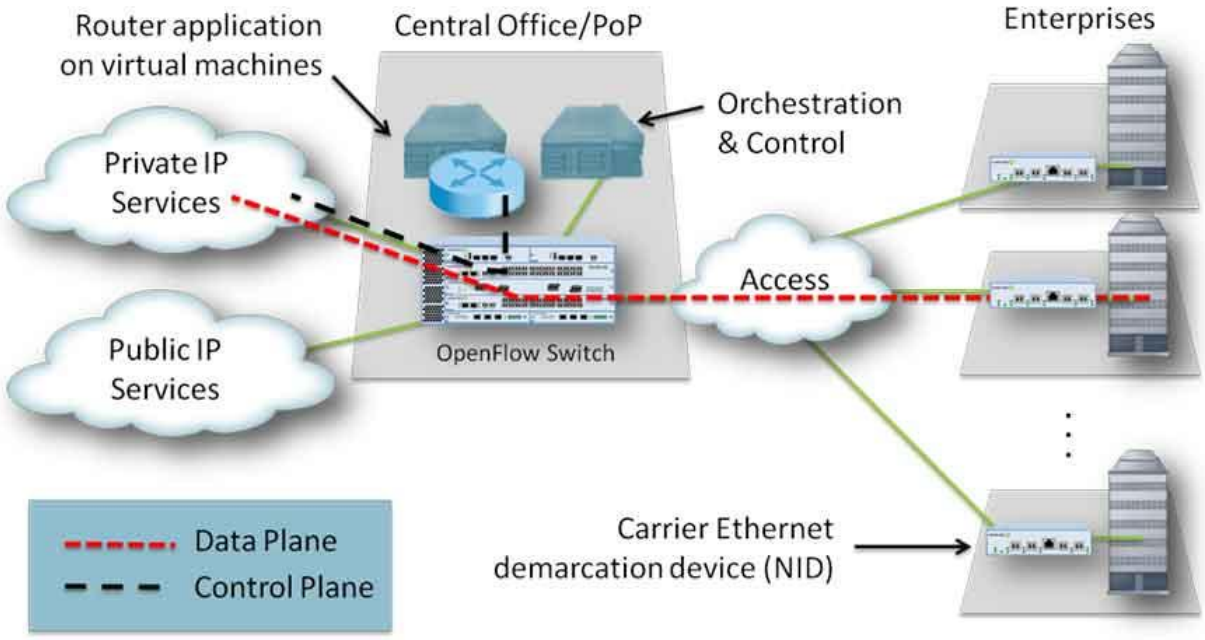


Рисунок 2.9 - Управління маршрутизацію з використанням NFV і SDN

SDN вводиться для відокремлення керування та даних, як показано на рисунку 2.9. Тепер пакети даних перенаправляються рівнем даних, тоді як функція маршрутизації (рівень керування) запускається у віртуальній машині. Таким чином комбінація NFV і SDN забезпечує оптимальні рішення:

- Дорогий окремий замінено загальним обладнанням та сучасним програмним забезпеченням;
- Рівень управління програмним забезпеченням переміщується зі спеціальної платформи в оптимізовану стандартну (сервер в центрі обробки даних);
- Управління рівнем даних було абстраговане та стандартизоване, що дозволило розвивати мережу та додатки без необхідності модернізації мережевих пристроїв.

У таблиці 2.1 наведені основні відмінності між SDN і NFV.

Таблиця 2.1 - Порівняння SDN і NVF

	SDN	**NFV**

Основна ідея	Розділення управління і даних, централізація управління	Винесення мережевих функцій з спеціальних приборів на загальні сервера
Цільове розташування	Хмари, центри даних	Мережа
Цільовий пристрій	Сервери і комутатори	Сервери і комутатори
Нові протоколи	OpenFlow	-

Також NFV і SDN повинні відіграти велику роль в формуванні 5G мереж. NFV допоможе віртуалізувати декілька пристроїв у мережі. Зокрема, NFV дозволить розрізати мережу 5G, дозволяючи різним віртуальним мережам працювати в єдиній фізичній інфраструктурі. Крім того, 5G NFV дозволить розділити фізичну мережу на різні віртуальні мережі, здатні підтримувати декілька мереж радіодоступу. NFV також може вирішити проблеми на шляху до 5G, оптимізувавши надання ресурсів віртуальних мережевих функцій.

SDN може використовуватися для забезпечення загальної структури, що дозволяє 5G функціонувати на рівні управління. Це може покращити передачу даних, коли вони проходять мережею 5G. Крім того, архітектура SDN може збільшити пропускну здатність мережі та зменшити затримку. Нарешті, оскільки SDN може використовуватися в мережах 5G, він забезпечує спосіб керування та автоматизації резервування мережі з центрального рівня управління, що обмежує основні переривання під час роботи, визначаючи оптимальні потоки даних у режимі реального часу.

2.2 Порівняння існуючих контролерів SDN

Як правило, SDN контролер складається з декількох частин:

1. Зовнішні API, що взаємодіють з модулями відповідальними за

моніторинг мережі, маршрутизацію, віртуалізацію, аутентифікацію та інші основні мережеві функції;

2. Внутрішні мережеві додатки та віртуальні сервери, що можуть містити бази даних та використовувати алгоритми пошуку оптимального шляху;
3. Інтерфейси OpenFlow, для взаємодії з рівнем даних, і розбиття процесів на декілька потоків.

Як правило, при проектуванні контролера звертають увагу на такі показники, як:

- пропускна спроможність;
- затримка;
- надійність;
- логічні інтерфейси.

Деякі виробники вважають, що для користування перевагами SDN не обов'язково використовувати OpenFlow. Продукти від Nicira, Juniper, Cisco та багатьох інших стартапів SDN не залежать від найнижчого рівня мережі та пропонують програмні рішення для масштабованої та віртуалізованої інфраструктури без OpenFlow. Такі продукти мають функції, які простіше реалізовувати для підприємств та хмарних сервісів. Цей підхід доцільний для тих підприємств, які мають ресурси для програмування та підтримки нового мережевого коду для нових маршрутизаторів.

Такий підхід також привабливий для тих компаній, які не хочуть замінювати існуючу базу обладнання для придбання нового мережевого пристрою на базі OpenFlow.

Прихильники патентів критикують підхід створення віртуального мережевого накладання (продукт NSX від VMware), який вільно з'єднується з фізичною мережею, не дивлячись на низьку масштабованістю. Доречність використання програмних комутаторів та накладання віртуальних мереж, яких буде достатньо для обробки інформації в високопродуктивних

середовищах, залежить від ситуації, а не від загальної масштабованості і продуктивності мережевого пристрою.

Прихильники відкритих платформ критикують виробників, які продовжують скривати власні компоненти у своєму продукті. Наприклад, багато критики було висловлено в адресу Cisco. Основні маршрутизатори розташовані в центрі даних і мають основний контроль над розподілом інформації. Дані переміщуються з ядра мережі до кінця рядка стійок в центр обробки даних, а потім - до перемикачів у верхній частині кожної стійки. Ядро є основним продуктом Cisco. Cisco має три різні операційні системи (IOS, IOS-XR і NX-OS), які реалізовані з використанням різних технологій. Ці операційні системи мають свої особливості реалізації, які необхідно враховувати при розробці мережевих додатків SDN. Як правило, такі особливості реалізації технології відштовхують потенційних користувачів SDN, оскільки складається враження, що перехід з традиційної організації мережі до SDN може зайняти багато часу і потребує заміни багатьох ресурсів мережі.

Незважаючи на цю критику, Cisco активно впроваджує концепцію SDN. Cisco може це зробити, оскільки багато традиційних мереж побудовані з використанням технологій Cisco, які можна повторно використовувати і для SDN мереж. Сьогоднішні мережеві пристрої виробляються з інкапсуляції Fibre Channel в пакети Ethernet (FCoE), щоб зменшити складність.

Завдяки Dynamic Fabric Automation (DFA), Cisco є єдиним постачальником на ринку, який використовує стратегію організації функцій фізичного тунелювання в мережевому обладнанні з мережевими агентами програмного забезпечення, такими як Nexus 1000V. Це дозволяє розгортати накладені мережі, які з'єднують обидві віртуалізовані платформи, такі як OpenStack або VMware, з фізичними пристроями та серверами. Замість підтримки віртуальних навантажень у хмарній платформі, наприклад, vCloud або OpenStack, пристрої Cisco можуть підтримувати як фізичне навантаження, так і віртуальне на фізичних і віртуальних пристроях. DFA

виглядає як надійний продукт, який, безумовно, відповідає потребам клієнтів, виходить за рамки конкурентної продукції та відтворює здатність Cisco інтегрувати фізичні та віртуальні мережі. На жаль, вибір нестандартних інтерфейсів та інкапсуляції критикується як значний недолік.

Хоча SDN є підходом, який повинен створити мережу центрів обробки даних нового покоління, він може виявитись зовсім невідповідним для цих цілей, оскільки системи SDN базуються на абстракції існуючих моделей мережі. Цей атрибут обмежує здатність SDN об'єднувати керування фізичними та віртуальними мережевими ресурсами. Фокусування SDN на всій мережі, а не на керуванні однією мережевою областю одночасно, є ще однією проблемою, яка обмежує можливості програмування та можливості застосовувати обмежень управління до всієї мережі з єдиної точки.

Багато компаній пов'язаних з мережевими технологіями створили свої реалізації контролерів SDN. Всі вони відповідають стандартним архітектурам контролерів і загальним принципам роботи, проте, між ними існують відмінності. По-перше, кожен виробник контролера намагається стандартизувати його під інші власні пристрої. Тому, реалізація деяких інтерфейсів може суттєво відрізнятися. Наприклад, Cisco розробила свій власний контролер для використання його в мережі побудованій за власними стандартами. По-друге, програмна частина різних контролерів реалізована на різних мовах програмування. На це вплинуло декілька факторів, одним з яких можна назвати фізичну реалізацію мережі. Окрім того, адміністратор мережі має можливість вибрати найбільш відповідний до його вимог контролер. Проте, це зовсім не впливає на мову на якій написані додатки, оскільки додатки знаходяться на іншому рівні архітектури SDN. Тож найбільш популярними реалізаціями контролера SDN є:

- Cisco Application Policy Infrastructure Controller (APIC) вважається розподіленою системою, реалізованою як кластер контролерів. В рамках програми Cisco Application Centric Infrastructure (ACI) APIC виступає в якості єдиної точки контролю. Він забезпечує центральний API, центральний

репозиторій глобальних даних і сховище даних. Контролер може автоматично застосовувати мережеві обмеження та функції, орієнтовані на додаток на основі моделі даних. Основна мета APIC полягає в тому, щоб забезпечити обмеження авторизації та регулювання мережевих обмежень для пристроїв ACI Cisco для оптимізації продуктивності та ефективності роботи мережі. Автоматизація проявляться як прямий результат регулювання обмежень та їх впливів на ACI.

Існують як основні ACI вузли, так і листові з якими APIC підтримує зв'язок. Таким чином, він може поширювати обмеження та виконувати ряд адміністративних функцій. Якщо безпосередньо контролер не бере участь у переадресації даних, кластер не втратить ніякої функціональності центру даних, при відключенні компонентів APIC.

- Контролер SDN HP Virtual Application Networks (VAN) контролює рішення щодо обмежень та пересилань в мережі SDN за допомогою комутаторів з підтримкою OpenFlow в центрі обробки даних або в інфраструктурі. HP також співпрацює з Verizon та Intel для розробки додатків, які використовуються для забезпечення пропускної спроможності WAN за допомогою контролера VAN;

Контролер також забезпечує централізоване управління та автоматизацію. В середовищі HP SDN контролер VAN забезпечує інтеграцію між мережею та логічною частиною системи. Він використовує програмовані інтерфейси, які дозволяють регулювати роботу додатка та автоматизувати мережеві функції. Контролер також забезпечує контроль над мережею, включаючи функції, такі як виявлення топології мережі;

З контролерів VAN також можна організувати кластер, що дозволяє контролеру виконувати функції іншого контролера, у разі виходу зі строю одного з контролерів. Для забезпечення безпеки контролер використовує методи аутентифікації та авторизації. У свою чергу, програми SDN можуть взаємодіяти з контролером, а неавторизовані програми не можуть отримати

доступ до мережі. Інтерфейси для зв'язку між перемикачами OpenFlow і контролером HP також захищені та зашифровані;

- Контролер NEC ProgrammableFlow PF6800 знаходиться в центрі мережі, заснованої на ProgrammableFlow OpenFlow NEC. Він забезпечує контроль фізичних мереж та управління і контроль віртуальних. Контролер є логічним елементом системи. Він інтегрується як з OpenStack, так і з Microsoft System Center Virtual Machine Manager для додаткового управління мережею та налаштування. Контролер також включає віртуальну мережеву технологію NEC, яка дозволяє використовувати ізольовані мережі;

- Nuage Networks Virtualized Services Controller (VSC) дозволяє переглядати топології мереж і служб для кожного власника, а також експортувати шаблони служб мережі, визначені за допомогою віртуалізованих каталогів Nuage Networks. Сервісний каталог - це механізм, який використовує мережеву аналітику та правила доступів на основі ролі. VSC надсилає повідомлення за допомогою цих правил на віртуальний маршрутизатор та комутатор платформи Nuage. Платформа управляє створенням або видаленням віртуальної машини, а потім запитує контролер SDN, якщо для цього користувача існують обмеження;

- Контролер VMware NSX є розподіленою системою управління, яка керує віртуальними мережами та транспортними тунелями. Це центральна контрольна точка для всіх логічних комутаторів всередині мережі. Контролер зберігає інформацію про віртуальні машини, хости, логічні комутатори та VXLAN, використовуючи високорівневий API для спілкування з додатками.

Під час роботи з контролером додатки повідомляють про те, що вони потребують, а контролер програмного забезпечення намагається налаштувати відповідну частину мережі під запити додатків. Контролер

може працювати двома шляхами в NSX: або як кластер віртуальних машин у середовищі vSphere, або як фізичні пристрої.

На рисунку 2.10 можна побачити порівняння продуктивності контролерів. В даному графіку зображена кількість потоків в секунду, яку може обробити кожен контролер відносно кількості віртуальних процесів. Також, на рисунку 2.11 можна побачити функціональність контролерів різних виробників.

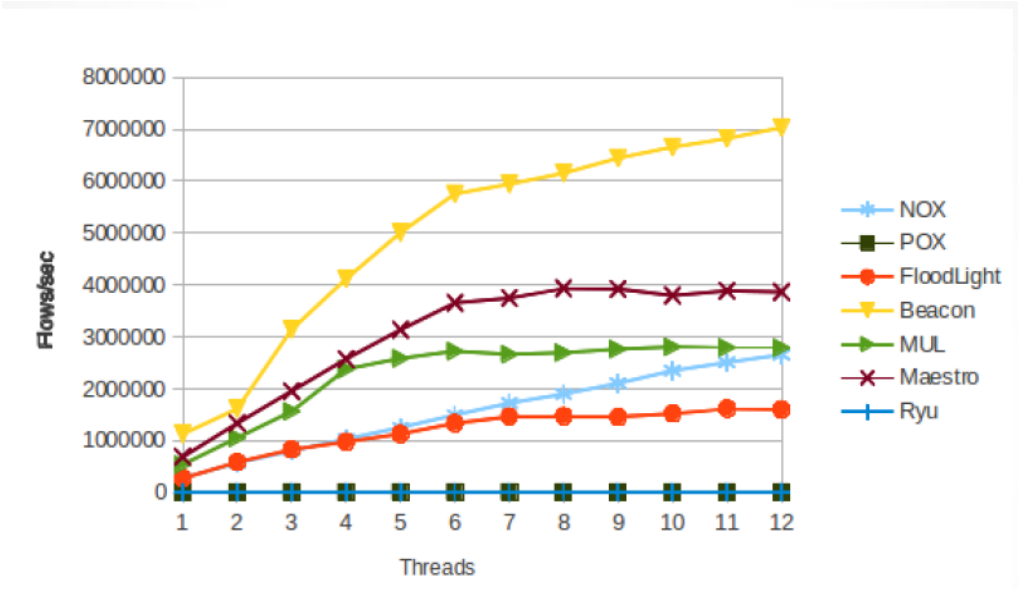


Рисунок 2.10 - Продуктивність контролерів різних виробників

Use-Cases \ Controllers	Trema	Nox/Pox	RYU	Floodlight	ODL	ONOS***
Network Virtualizaiton by Virtual Overlays	YES	YES	YES	PARTIAL	YES	NO
Hop-by-hop Network Virtualization	NO	NO	NO	YES	YES	YES
OpenStack Neutron Support	NO	NO	YES	YES	YES	NO
Legacy Network Interoperability	NO	NO	NO	NO	YES	PARTIAL
Service Insertion and Chaining	NO	NO	PARTIAL	NO	YES	PARTIAL
Network Monitoring	PARTIAL	PARTIAL	YES	YES	YES	YES
Policy Enforcement	NO	NO	NO	PARTIAL	YES	PARTIAL
Load Balancing	NO	NO	NO	NO	YES	NO
Traffic Engineering	PARTIAL	PARTIAL	PARTIAL	PARTIAL	YES	PARTIAL
Dynamic Network Taps	NO	NO	YES	YES	YES	NO
Multi-Layer Network Optimization	NO	NO	NO	NO	PARTIAL	PARTIAL
Transport Networks - NV, Traffic-Rerouting, Interconnecting DCs, etc.	NO	NO	PARTIAL	NO	PARTIAL	PARTIAL
Campus Networks	PARTIAL	PARTIAL	PARTIAL	PARTIAL	PARTIAL	NO
Routing	YES	NO	YES	YES	YES	YES

Рисунок 2.11 - Функціональність контролерів різних виробників

2.3 Основні проблеми

SDN розділяє рівень управління і рівень даних, що призводить до значного спрощення модуля управління. Таким чином вся логіка управління мережею зосереджена, наприклад, на одному сервері, а інші пристрої мережі, відповідно до концепції, використовуються тільки для передачі даних. Проте одним із основних недоліків нової архітектури є складність масштабування рівня управління мережі. Зі збільшенням кількості вузлів у мережі, і, відповідно, зі значним збільшенням запитів до рівня управління, навантаження на модуль управління зростає, і його продуктивність може значною мірою знижуватися. Таким чином, обчислення масштабованості рівня управління SDN є критичною проблемою для успішної адаптації SDN до великомасштабних мереж або мереж з великою кількістю потоків.

Оскільки мережі SDN використовують новий принцип побудови мереж, переважна більшість існуючих досліджень не створюють модель, а базуються на експериментах і моделюванні завдяки спеціалізованому програмному забезпеченню. Метою роботи є математичне порівняння різних типів архітектури рівня управління SDN.

Незважаючи на переваги SDN, технологія досі потребує оптимізації та вирішення багатьох проблем, до яких відносяться наступні:

Динамічно адресувати зміни у реальному часі. Можливість автоматизації об'єднання нових інфраструктур протягом декількох хвилин з впливом на декілька пристроїв одночасно є складною задачею, особливо з огляду на те, що відносні статичні середовища сьогодні залежать від ручних конфігурацій. Завдяки SDN, нові обчислювальні пристрої, мережеві пристрої та пристрої для зберігання інформації можуть бути відразу доступні для використання. Щоденні перевірки на наявність нових пристроїв в мережі залишають величезні прогалини у загальній видимості мережі.

Для вирішення цієї проблеми, необхідно розробити інструмент для моніторингу змін у мережі, спроектований за допомогою відкритих API.

Після цього, адміністратору чи оператору мережі не потрібно буде перевіряти наявність нових пристроїв вручну - кожен новий пристрій буде автоматично додаватися до мережі.

Забезпечення швидкого зростання. Зростаючий попит на нові обчислення, мережі та сховища в SDN ускладнює моніторинг платформи. На рисунку 2.12. можна побачити прогноз по зростанню мереж SDN. Необхідне рішення, що значно збільшить потужності моніторингу для швидкого зростання інфраструктури. Якщо вони не можуть додати додаткову потужність за вимогою, вони почнуть втрачати продуктивність, створюючи пробіли у моніторингу.

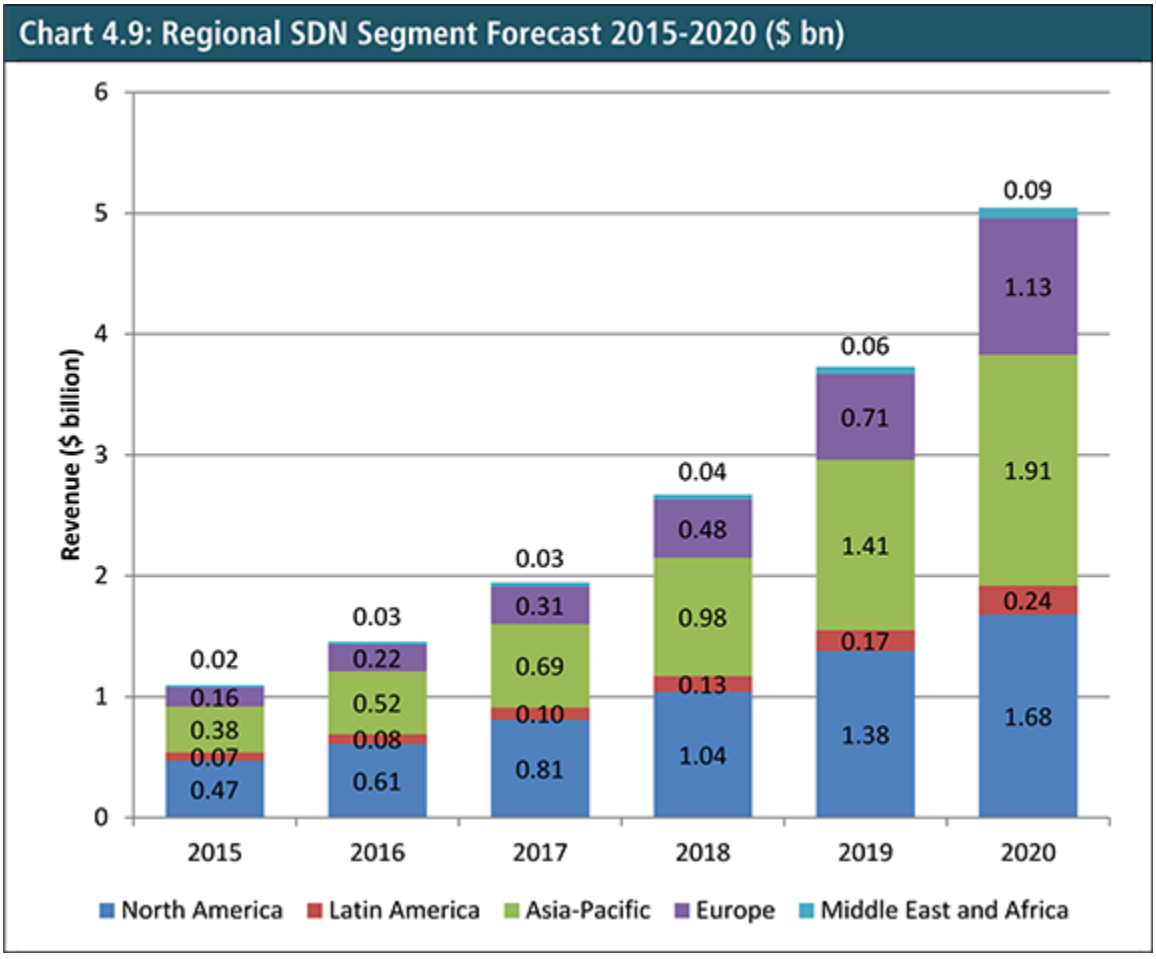


Рисунок 2.12 - Прогноз розвитку мереж SDN

На відміну від застарілих інфраструктур у світі SDN, ми можемо мати декілька накладених топологій, що працюють на вершині фізичної мережі.

Кожного разу, коли запускається новий сервіс, розгортається необхідна віртуальна інфоструктура, і, отже, кількість контрольованих елементів може швидко зростати з підвищеним попитом, перевершуючи традиційний менеджмент потенціалу.

Рішенням є розгортання моніторингу продуктивності як у фізичному, так і у віртуальному вигляді. Коли потрібна додаткова продуктивність управління, необхідне збільшення кількості віртуальних пристроїв за замовчуванням дозволяє контролювати продуктивність з вимогами середовища SDN та надавати відповіді впродовж декількох секунд.

Інтеграція контексту сервісу. В результаті моніторинг продуктивності повинен мати можливість зчитувати контекст конкретного клієнта або орендаря мережі. Зрештою, користувачі повинні мати змогу не лише запитати про стан та продуктивність окремих пристроїв або посилань у мережі, а й про те, як саме працює клієнт.

Це також поширюється на топологію обслуговування, тобто контролери та рішення для моніторингу продуктивності обмінюються інформацією про фізичні і логічні підключення.

SDN все ще дозріває, і в процесі його еволюції важливо подивитися, як динамічні зміни в реальному часі, швидке зростання за вимогою та інтеграція контексту послуг відіграватимуть ключову роль у забезпеченні успішного розгортання та уникненні прогалин у видимості діяльності у вашій інфраструктурі.

Проектування SDN відрізняється від проектування традиційних мереж і протоколів, так як мережі SDN за своєю природою є орієнтованими на сервіси і повинні гнучко змінювати значення параметрів і характеристик в реальному часі, а також враховувати асинхронний режим взаємодії елементів і можливість множинного доступу до ресурсів мережі. В якості моделі життєвого циклу SDN застосовна спіральна модель. Вона успадковує всі переваги каскадної моделі, її характерною особливістю є можливість внесення змін на ранніх стадіях розробки і завершення кожного етапу

перевіркою отриманих результатів, що забезпечує істотне зниження тимчасових і матеріальних витрат. Однак повноцінне функціонування подібної мережі на сьогоднішній день важко організувати. Основні проблеми, що виникають в процесі функціонування мереж, побудованих на основі концепції SDN, пов'язані з відсутністю стандартизованих протоколів взаємодії, наявністю неточностей і розбіжностей в існуючих версіях специфікацій.

Відсутність єдиних стандартів також істотно впливає і на проектування SDN: організатори повинні враховувати, самостійно формалізувати і впроваджувати в свої рішення всі вимоги, закладені в різних версіях OpenFlow протоколів.

Таким чином, на різних етапах життєвого циклу SDN можливе виникнення ряду помилок, причинами яких є:

- неповний аналіз предметної області;
- помилки, що виникають при формалізації специфікації;
- різна інтерпретація стандартів;
- логічні помилки проектування;
- неповна відповідність пропонованим вимога;
- неправильне урізання функціональності протоколу;
- надмірність в реалізації протоколу;
- помилки при реалізації протоколу.

Для усунення помилок, можна сформувати ряд вимог, які можна застосувати на кожному етапі життєвого циклу протоколів SDN, які дозволять уникнути помилок і підвищують ефективність функціонування мережі:

- формальний опис вимог, що пред'являються до додатків SDN, враховуючи особливості середовища і характеру переданих даних;
- розробка формальних методів перевірки спільного функціонування контролерів різних виробників: перевірка несуперечності команд і правил, закладених в логіку роботи контролера;

- розробка формальних методів перевірки взаємодії між рівнем додатків і передачею даних: перевірка коректності виконання команд і їх несуперечливість;
- формальне уніфіковане представлення специфікацій, які забезпечать єдиний підхід до взаємодії контролерів і додатків, представлених різними розробниками;
- розробка формальних методів перевірки відповідності готового рішення SDN його специфікації.

Здійснення всіх перерахованих кроків можливо в процесі верифікації на кожному етапі розробки. При такому підході можливе істотне скорочення вартості розробки і впровадження нових додатків, а також придбання прогностичних знань про взаємодію між рівнями і створення шаблонних підходів.

Ще одною проблемою на шляху до впровадження SDN є трансформація мережі стандартного типу до SDN. Оскільки, як правило, SDN використовує протокол OpenFlow для обміну запитами між рівнем даних, це може ввести деяке обмеження на використання пристроїв. Тому одним з основних факторів при переході на SDN є впровадження в мережу OpenFlow пристроїв, які будуть взаємодіяти з існуючими пристроями. Крім того, необхідно обміркувати, яким чином буде здійснене повернення до старого типу архітектури, оскільки, після переходу на SDN деякі функції мережі можуть перестати працювати. Також, важливим є перехід з фізичних серверів і пристроїв до віртуальних.

Висновки до розділу 2

Проведено детальний аналіз різних типів архітектури рівня управління. Проаналізована роль кожного елемента, що входить до рівня управління. Освітлено і проаналізовано найпоширеніші проблеми рівня управління і SDN мереж в цілому.

Проведено аналітичне порівняння декількох типів контролерів від різних виробників. Показана різниця між SDN і NFV.

Показано, що на сьогодні хоча і розроблені декілька видів архітектур рівня управління, та залишається невирішеним питання масштабованості і швидкодії мережі. Крім того, необхідно приділяти увагу безпеці рівня управління.

3. МОДИФІКОВАНА АРХІТЕКТУРА SDN

3.1 Критерії оцінки

Не дивлячись на те, що частина компанії вже впровадила SDN в організацію своїх мереж, технологія досі залишається на стадії вивчення і усунення недоліків, тому більша частина потенціальних користувачів не використовують цей підхід.

Оскільки SDN є підходом до проектування мереж, то і критерії оцінки використовують, як правило, такі ж як і до інших видів мереж.

Головною вимогою до мережі є виконання мережею її основної функції - забезпечення користувачів потенційною можливістю доступу до ресурсів всіх комп'ютерів, об'єднаних в мережу. Всі інші вимоги - продуктивність, надійність, сумісність, керованість, захищеність, розширюваність і масштабованість - пов'язані з якістю виконання цієї основної задачі.

Хоча всі ці вимоги дуже важливі, часто поняття «якість обслуговування» комп'ютерної мережі трактується більш вузько: в нього включаються тільки дві найважливіші характеристики мережі - продуктивність і надійність.

Незалежно від обраного показника якості обслуговування мережі існують два підходи до його забезпечення. Перший підхід полягає в тому, що мережа гарантує користувачеві дотримання деякої числової величини показника якості обслуговування. Наприклад, мережа може гарантувати користувачу А, що будь-який з його пакетів, посланих користувачеві В, буде затриманий мережею не більше, ніж на 150 мілісекунд. Або, що середня пропускна спроможність каналу між користувачами А і В не буде нижче 5 мегабіт в секунду, при цьому канал буде дозволяти пульсації трафіку в 10 Мбіт на інтервалах часу не більше 2 секунд. Технології frame relay і ATM дозволяють будувати мережі, що гарантують якість обслуговування по продуктивності.

Другий підхід полягає в тому, що мережа обслуговує користувачів відповідно до їх пріоритетів. Тобто якість обслуговування залежить від

ступеня привілейованості користувача або групи користувачів, до якої він належить. Якість обслуговування в цьому випадку не гарантується, а гарантується тільки рівень привілеїв користувача. За таким принципом працюють, наприклад, локальні мережі, побудовані на комутаторах з пріоритезацією кадрів.

Розглянемо основні критерії оцінки мережі.

Продуктивність. Існує кілька основних характеристик продуктивності мережі: час реакції, пропускна спроможність, затримка при передачі.

Час реакції мережі є інтегральною характеристикою продуктивності мережі з точки зору користувача. У загальному випадку час реакції визначається як інтервал часу між виникненням запиту користувача до якої-небудь мережевої служби і отриманням відповіді на цей запит.

Тому має сенс використовувати також і середньозважену оцінку часу реакції мережі, беручи середній показник по користувачах, серверах і часу дня (від якого в значній мірі залежить завантаження мережі).

Час реакції мережі зазвичай складається з декількох складових. У загальному випадку в нього входить час підготовки запитів на клієнтському комп'ютері, час передачі запитів між клієнтом і сервером через сегменти мережі і проміжне комунікаційне обладнання, час обробки запитів на сервері, час передачі відповідей від сервера клієнту і час обробки одержуваних від сервера відповідей на клієнтському комп'ютері .

Знання складових часу реакції дає можливість оцінити продуктивність окремих елементів мережі, виявити вузькі місця і в разі необхідності виконати модернізацію мережі для підвищення її загальної продуктивності.

Пропускна спроможність відображає обсяг даних, переданих мережею чи її частиною за одиницю часу. Пропускна спроможність вже не є для користувача характеристикою, так як вона говорить про швидкість виконання внутрішніх операцій мережі - передачі пакетів даних між вузлами мережі через різні комунікаційні пристрої. Пропускна спроможність безпосередньо характеризує якість виконання основної функції мережі -

транспортування повідомлень - і тому частіше використовується при аналізі продуктивності мережі, ніж час реакції.

Пропускна спроможність вимірюється або в бітах в секунду, або в пакетах в секунду. Пропускна спроможність може бути миттєвою, максимальною і середньою.

Середня пропускна спроможність обчислюється шляхом ділення загального обсягу переданих даних на час їх передачі, причому вибирається досить тривалий проміжок часу - годину, день або тиждень.

Миттєва пропускна спроможність відрізняється від середньої тим, що для усереднення вибирається дуже маленький проміжок часу, наприклад, 10 мілісекунд або 1 секунда.

Максимальна пропускна спроможність - це найбільша миттєва пропускна спроможність, зафіксована протягом періоду спостереження.

Найчастіше при проектуванні, налаштуванні і оптимізації мережі використовуються такі показники, як середня і максимальна пропускні спроможності.

Середня пропускна спроможність окремого елемента або всієї мережі дозволяє оцінити роботу мережі на великому проміжку часу.

Максимальна пропускна спроможність дозволяє оцінити можливості мережі справлятися з піковими навантаженнями, характерними для особливих періодів роботи мережі.

Через послідовний характер передачі пакетів різними елементами мережі, загальна пропускна спроможність мережі будь-якого складеного шляху в мережі буде дорівнює мінімальній з пропускних спроможностей складових елементів маршруту.

Загальна пропускна спроможність мережі визначається як середня кількість інформації, переданої між всіма вузлами мережі в одиницю часу. Цей показник характеризує якість мережі в цілому, не диференціюючи його по окремих сегментах або пристроях.

Зазвичай при визначенні пропускної спроможності сегмента або пристрою в даних, що передаються не виділяються пакети якогось певного користувача, додатка або комп'ютера - підраховується загальний обсяг переданої інформації. Однак для більш точної оцінки якості обслуговування така деталізація бажана, і, останнім часом, системи управління мережами все частіше дозволяють її виконувати.

Затримка передачі визначається як затримка між моментом надходження пакету на вхід якого-небудь мережевого пристрою або частини мережі і моментом появи його на виході цього пристрою.

Не всі типи трафіка чутливі до затримок передачі, які характерні для комп'ютерних мереж (зазвичай затримки не перевищують сотень мілісекунд, рідше - кількох секунд). Такого порядку затримки пакетів, породжуваних файлової службою, службою електронної пошти або службою друку, мало впливають на якість цих служб з точки зору користувача мережі. З іншого боку, такі ж затримки пакетів, що переносять голосові дані або відео, можуть призводити до значного зниження якості інформації.

Пропускна спроможність і затримки передачі є незалежними параметрами, так що мережа може мати, наприклад, високу пропускну спроможність, але вносити значні затримки при передачі кожного пакета.

Надійність. Для технічних пристроїв використовуються такі показники надійності, як середній час напрацювання на відмову, імовірність відмови, інтенсивність відмов. Однак ці показники придатні для оцінки надійності простих елементів і пристроїв, які можуть перебувати лише в двох станах - працездатному або непрацездатному. Складні системи, що складаються з багатьох елементів, крім станів працездатності та непрацездатності, можуть мати і інші проміжні стани, які ці характеристики не враховують. У зв'язку з цим для оцінки надійності складних систем застосовується інший набір характеристик.

Готовністю або коефіцієнтом готовності називають проміжок часу, протягом якого система може бути використана. Готовність може бути

поліпшена шляхом введення надмірності в структуру системи: ключові елементи системи повинні існувати в декількох екземплярах, щоб при відмові одного з них функціонування системи забезпечували інші.

Крім того, необхідно забезпечити збереження даних, захист їх від спотворення або несуперечливість даних (наприклад, якщо для підвищення надійності на декількох файлових серверах зберігається кілька копій даних, то потрібно постійно забезпечувати їхню ідентичність).

Ще однією характеристикою надійності є відмовостійкість. У мережах під відмовостійкісттю розуміється здатність системи приховати від користувача відмову окремих її елементів. Наприклад, якщо копії таблиці бази даних зберігаються одночасно на декількох файлових серверах, то користувачі можуть просто не помітити відмову одного з них.

Безпекою називають здатність системи захистити дані від несанкціонованого доступу. У розподіленій системі це зробити набагато складніше, ніж в централізованій.

У мережах повідомлення передається по лініях зв'язку, що часто проходять через загальнодоступні приміщення, в яких можуть бути встановлені засоби прослуховування ліній. Іншим вразливим місцем можуть бути залишені без нагляду персональні комп'ютери. Крім того, завжди є потенційна загроза злому захисту мережі, якщо мережа має виходи в глобальні мережі загального користування.

Можливість розширення і масштабованості. Терміни розширюваність і масштабованість іноді використовують як синоніми, але це невірно - кожен з них має чітко визначене самостійне значення.

Можливість розширення означає можливість порівняно легкого додавання окремих елементів мережі (користувачів, комп'ютерів, додатків, служб), нарощування довжини сегментів мережі і заміни існуючої апаратури більш потужною. При цьому принципово важливо, що легкість розширення системи іноді може забезпечуватися в деяких дуже обмежених межах.

Наприклад, локальна мережа Ethernet, побудована на основі одного сегмента товстого коаксіального кабелю, має гарну розширюваністю, в тому сенсі, що дозволяє легко підключати нові станції. Однак така мережа має обмеження на кількість станцій (їх число не повинно перевищувати 30-40). Наявність такого обмеження і є ознакою поганої масштабованості системи при хорошій розширюваності.

Масштабованість означає, що мережа дозволяє нарощувати кількість вузлів і протяжність зв'язків в дуже широких межах, при цьому продуктивність мережі не погіршується.

Для забезпечення масштабованості мережі доводиться застосовувати додаткове комунікаційне обладнання і спеціальним чином структурувати мережу.

Наприклад, хорошою масштабністю володіє багатосегментна мережа, побудована з використанням комутаторів і маршрутизаторів і має ієрархічну структуру зв'язків. Така мережа може включати кілька тисяч комп'ютерів і при цьому забезпечувати кожному користувачеві мережі потрібну якість обслуговування.

Прозорість мережі досягається в тому випадку, коли мережа представляється користувачам не як безліч окремих комп'ютерів, зв'язаних між собою складною системою кабелів, а як єдина традиційна обчислювальна машина. Навіть існує відоме гасло компанії Sun Microsystems: «Мережа - це комп'ютер», що говорить саме про таку прозору мережу.

На рівні користувача прозорість означає, що для роботи з віддаленими ресурсами він використовує ті ж команди і звичні йому процедури, що і для роботи з локальними ресурсами. Прозорість на рівні додатку вимагає приховання всіх деталей розподіленості засобами мережної операційної системи.

Мережа повинна приховувати всі особливості операційних систем і відмінності в типах комп'ютерів. Користувач UNIX повинен мати можливість розділяти інформацію з користувачами Windows і навпаки.

Концепція прозорості може бути застосована до різних аспектів мережі. Наприклад, прозорість розташування означає, що від користувача не потрібно знань про місце розташування програмних і апаратних ресурсів, таких як процесори, принтери, файли і бази даних. Ім'я ресурсу не повинне включати інформацію про місце його розташування.

В даний час не можна сказати, що властивість прозорості в повній мірі властиво багатьом обчислювальним мережам, це скоріше мета, до якої прагнуть розробники сучасних мереж.

Підтримка різних видів трафіка. Комп'ютерні мережі, призначені для спільного доступу користувача до ресурсів комп'ютерів: файлам, принтерам тощо.

Однак 90-і роки стали роками проникнення в комп'ютерні мережі трафіку мультимедійних даних, що представляють в цифровій формі голос і відеозображення. Комп'ютерні мережі стали використовуватися для організації відеоконференцій, навчання і розваг на основі відеофільмів тощо. Природно, що для динамічної передачі мультимедійного трафіка потрібні інші алгоритми і протоколи і, відповідно, інше обладнання.

Головною особливістю трафіку, що утворюється при динамічній передачі голосу або зображення, є наявність жорстких вимог до синхронності повідомлень. При запізненні повідомлень будуть спостерігатися спотворення.

У той же час трафік комп'ютерних даних характеризується вкрай нерівномірною інтенсивністю надходження повідомлень в мережу при відсутності жорстких вимог до синхронності доставки цих повідомлень (наприклад, редагування тексту на віддаленому носії).

Всі алгоритми комп'ютерного зв'язку, відповідні протоколи і комунікаційне обладнання були розраховані саме на такий "пульсуючий"

характер трафіку, тому необхідність передавати мультимедійний трафік вимагає внесення принципових змін, як в протоколи, так і в обладнання. Сьогодні практично всі нові протоколи в тій чи іншій мірі надають підтримку мультимедійного трафіку.

Особливу складність представляє поєднання в одній мережі традиційного комп'ютерного та мультимедійного трафіку. Передача виключно мультимедійного трафіка комп'ютерною мережею хоч і пов'язана з певними складнощами, але викликає менші труднощі. А ось випадок співіснування двох типів трафіку з протилежними вимогами до якості обслуговування є набагато більш складним завданням.

Зазвичай протоколи і обладнання комп'ютерних мереж відносять мультимедійний трафік до факультативного (додаткового), тому якість його обслуговування залишає бажати кращого. Сьогодні витрачаються великі зусилля по створенню мереж, які не обмежують інтереси одного з типів трафіку. Найбільш близькі до цієї мети мережі на основі технології ATM, розробники якої спочатку враховували випадок співіснування різних типів трафіка в одній мережі.

Керованість мережі має на увазі можливість централізовано контролювати стан основних елементів мережі, виявляти і вирішувати проблеми, що виникають при роботі мережі, виконувати аналіз продуктивності і планувати розвиток мережі. В ідеалі засоби управління мережами являють собою систему, яка здійснює спостереження, контроль і управління кожним елементом мережі - від найпростіших до найскладніших пристроїв, при цьому така система розглядає мережу як єдине ціле, а не як розрізнений набір окремих пристроїв.

Хороша система управління спостерігає за мережею, і, виявивши проблему, активізує певні дії, виправляє ситуацію і повідомляє адміністратора про те, що сталося і які кроки зроблені. Одночасно з цим система управління повинна накопичувати дані, на підставі яких можна планувати розвиток мережі.

В даний час в області систем управління мережами багато невирішених проблем. Явно недостатньо дійсно зручних, компактних і багато протокольних засобів управління мережею. Більшість існуючих засобів зовсім не керують мережею, а всього лише здійснюють спостереження за її роботою. Вони стежать за мережею, але не виконують активних дій, якщо з мережею щось відбулося або незабаром відбудеться. Мало масштабованих систем управління, здатних обслуговувати як мережі масштабу відділу, так і мережі масштабу підприємства.

В SDN відокремлюють цілий рівень для цих цілей, тому контроль за мережею в даному підході є значно ефективнішим. Контролер є одним з ключових елементів архітектури SDN. Це єдина точка управління мережею, він визначає політики конфіденційності, конфігурацію. Взаємодія між вузлами мережі і контролером може відбуватися за рахунок різних протоколів. Деякі з них відкриті, деякі запатентовані (такі як OpenFlow, OVSDB тощо). Контролер може виглядати як єдиний вузол, так і кластер серверів. Реалізація децентралізованої архітектура контролерів може суттєво відрізнятися. Вона може бути як просто розподілена, так і ієрархічна.

Сумісність або інтегрованість означає, що мережа здатна включати в себе найрізноманітніше програмне й апаратне забезпечення, тобто в ній можуть співіснувати різні операційні системи, що підтримують різні стеки комунікаційних протоколів, і працювати апаратні засоби і додатки від різних виробників. Мережа, що складається з різнотипних елементів, називається неоднорідною чи гетерогенної, а якщо гетерогенна мережа працює без проблем, то вона є інтегрованою. Основний шлях побудови інтегрованих мереж - використання модулів, виконаних відповідно до відкритих стандартів і специфікацій.

При проектуванні мережі SDN, важливим питанням є наскільки сумісним є SDN рішення з іншими додатками? Чи можуть вони легко інтегруватися? Якщо так, то які опубліковані API-інтерфейси можна використати для інтеграції? Чи дозволяє архітектура SDN інтегруватися з

іншими хмарними додатками/інструментами керування, щоб забезпечити інтегрований підхід?

Питання пов'язані з роботою з хмарними сервісами є одними з найбільш важливих. Тому, в практичній реалізації, контролер SDN часто інтегрується з такими інструментами, як VMWare vCenter, OpenStack, CloudStack тощо. Це ключова вимога для забезпечення синхронізації стану між усіма інструментами, що складають загальне рішення.

Ще одним важливим аспектом є експлуатація простота. При правильній побудові мережі SDN, користувач повинен однаково легко управляти мережею через графічний інтерфейс, API або командну консоль.

Також важливо реалізувати інструмент для збору аналітичних даних. Аналітика - це не просто статистичні дані про кількість пакетів; більшість постачальників, надають графіки, що відображають статистику підрахунку пакетів; Аналітика повинна бути спроможна передбачити затримку, визначити втрати пакетів, передбачити проблему, яка може стати основною. Аналітика повинна мати можливість контролювати поведінку програм у всій мережі SDN.

3.2 Модифікація існуючої архітектури

Існують два загальноприйнятих варіанти поліпшення продуктивності рівня управління SDN. Перший – це перенести частину функції управління на інші пристрої в мережі, наприклад, мережеві комутатори. Цей підхід дійсно дозволяє значно зменшити навантаження на центральний вузол, і значно збільшує загальну продуктивність мережі. Проте, комутатори повинні бути побудовані на спеціальній інтегральній схемі і мати центральний процесор. Таким чином, збільшується складність розгортання мережі. Другий підхід полягає в проектуванні особливого розподіленого рівня управління, який передбачає розподілення навантаження між декількома контролерами. Іншими словами, рівень управління буде працювати як розподілена комп'ютерна система, що колективно обробляє вхідні мережеві запити.

Окрім того, ще одним аспектом, який значно впливає на продуктивність мережі в цілому є взаємодія контролера з OpenFlow комутаторами. Використання декількох потоків і управління завантаженістю шляхів маршрутизації є найбільш ресурсоємкими задачами. Окрім того, багато часу потребує отримання і обробка пакетів з каналу.

Як правило, архітектура рівня управління SDN може бути централізованою і децентралізованою. Децентралізована, в свою чергу, поділяється на два підвиди: локальний і глобальний. Вузли локального виду не бачать мережу в цілому. Кожен вузол обробляє запити тільки від частини вузлів мережі. В свою чергу, вузли в децентралізованих системах глобального виду можуть приймати запити від усіх вузлів мережі. Децентралізовані структури також називають розподіленими структурами.

Для розрахунків масштабованості рівня управління SDN можна використовувати метрики масштабованості для розподіленої системи. Масштабованість для розподілених систем базується на продуктивності. Розподілену систему можна назвати масштабованою, якщо продуктивність системи залишається на одному рівні при збільшенні розміру системи. Для рівня управління SDN продуктивність $F(N)$ може бути визначена як:

$$F(N) = \phi(N) \times \frac{T(N)}{C(N)},$$

Де: N – кількість вузлів в мережі, $\phi(N)$ – пропускна спроможність рівня управління в обробці мережевих запитів, $T(N)$ – середній час відгуку на кожен запит, $C(N)$ – вартість розгортання рівня управління (наприклад, вартість одного контролеру).

Таким чином, масштабованість для рівня управління SDN, розмір якої змінюється від N_2 до N_1 визначається як:

$$\Psi(N_1, N_2) = \frac{F(N_2)}{F(N_1)}.$$

В SDN існує декілька типів мережевих запитів, які рівень управління повинен обробляти. Наприклад, запит для спостерігання за станом мережі, оновлення стану мережі, відновлення після збою. Однак ініціювання потоку є

головною і основною функціональністю контролеру SDN. Тому основна увага приділяється обробці ініціювання потоку в рівні управління. В децентралізованих структурах локального вигляду та ієрархічних структурах обробка запиту ініціювання потоку може оброблятися декількома контролерами. Необхідно ввести такий структурний параметр, як середня дистанція контролерів, що визначає яка кількість контролерів в середньому необхідна для обробки запиту ініціювання потоку.

Навантажувальна спроможність контролера визначається центральним процесором, використанням пам'яті і завантаженістю мережі. Як правило, "вузьким місцем" є процесор. Тому будемо приймати до уваги тільки час, який витрачає центральний процесор для обробки одного запиту. Час обробки залежить від топології мережі, використаних алгоритмів маршрутизації і обчислювальної потужності контролера. Часова складність алгоритмів маршрутизації позначається $g(V,E)$, де V – кількість мережевих вузлів, а E – кількість мережевих з'єднань. Тоді ми припустимо, що час обробки підлягає експоненційному розподілу з середнім значенням $\frac{g(V,E)}{K}$, де K – обчислювальна потужність. Тому припустимо, що контролер використовує алгоритм Дейкстри для пошуку найкращого шляху. Складність алгоритму Дейкстри залежить від реалізації, і у загальному випадку складає $O(V^2)$. Для спрощення обчислень, в цій роботі допустимо, що $g(V,E) = V^2$. Оскільки надходження запитів з одного вузла до іншого піддається Пуассонівському розподіленню, а сума незалежних випадкових величин Пуассонівського розподілення представляє собою Пуассонівське розподілення, то сукупність запитів, що надходять до контролера, також піддаються Пуассонівському розподіленню. Таким чином, кожен контролер є моделлю черги M/M/1.

Спочатку визначимо час відгуку контролера в централізованій структурі при кількості вузлів N та базовій середній інтенсивності надходження запитів λ.

$$\lambda_C = N \times (N-1) \times \lambda .$$

Ця величина має пуассонівський розподіл. Час обробки запиту розподіляється за експоненціальним законом. І середній час обробки запиту $\mu_C(N)$ обернено пропорційний до масштабу мережі N. З цього випливає, що

$$\mu_C = \frac{K}{g(N)}.$$

Тоді середній час відгуку контролера при централізованій структурі буде:

$$E\{T_C(N)\} = \frac{1}{\mu_C - \lambda_C}.$$

В децентралізованій структурі декілька контролерів. Припустимо, що кількість контролерів буде m_D. В першому типі децентралізованної структури кожен контролер має свою локальну мережу. Тому, існують два випадки: коли весь шлях управляється одним контролером, і коли необхідно задіяти декілька контролерів. Схематичне представлення децентралізованої локальної структури можна побачити на рисунку 3.1.

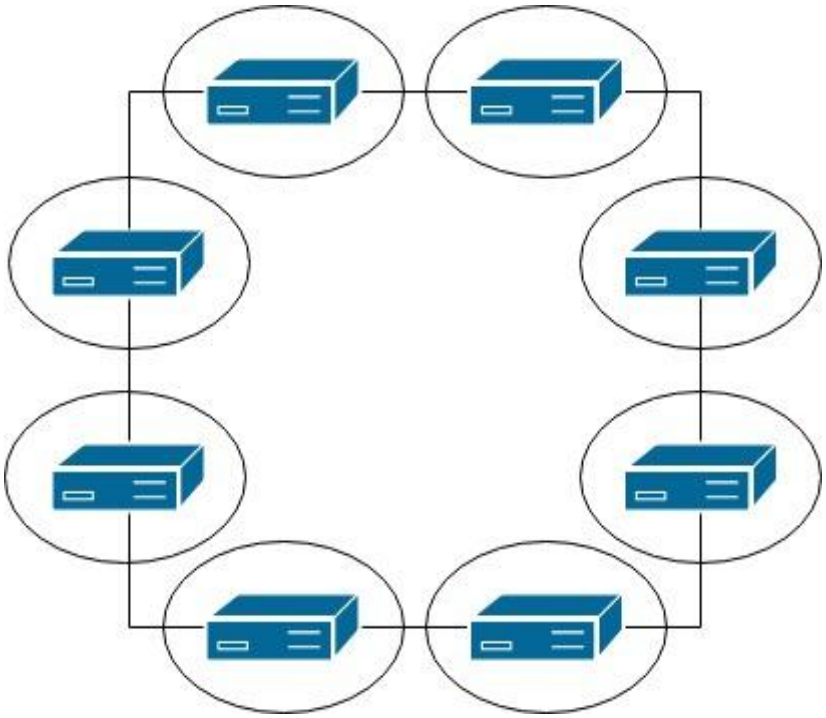


Рисунок 3.1 - Децентралізована локальна структура рівня управління

Глобальний запит поділяється на два запити: на локальний і на ще один локальний або глобальний до іншого контролера. Якщо середня дистанція

контролерів d_D, кожен глобальний запит буде поділений на $d_D + 1$ запитів. І час відгуку на глобальний запит буде дорівнювати сумі часу відгуку на локальні запити. Відповідно до цього, $\frac{N^2}{m_D} - N$ – кількість локальних шляхів, $N^2 - \frac{N^2}{m_D}$ – кількість глобальних шляхів. Тому швидкість прибуття запитів на кожен контролер буде:

$$\lambda_{D,l} = \lambda \times \frac{(N^2 - \frac{N^2}{m_D}) \times (d_D + 1) + (\frac{N^2}{m_D} - N)}{m_D},$$

а також кожен контролер повинен керувати топологією з $\frac{N}{m_D} + m_D - 1$ вузлами. Тому середній час обробки запиту:

$$\mu_{D,l} = \frac{K}{g(\frac{N}{m_D} + m_D - 1)},$$

а середній час відгуку контролера:

$$E\{T_{D,l}\} = \frac{1 + \frac{N \times (m_D - 1)}{(N-1) \times m_D} \times d_D}{\mu_{D,l} - \lambda_{D,l}}.$$

В другому типі децентралізованної структури, кожен вузол контролеру має повний доступ до мережі. Схематичне представлення децентралізованої глобальної структури можна побачити на рисунку 3.2. Ця властивість дозволяє розрахувати середній час обробки запиту і середній час відгуку контролера за формулами для централізованої структури. Швидкість прибуття запитів на кожен контролер буде:

$$\lambda_{D,g} = \frac{N \times (N-1) \times \lambda}{m_D}.$$

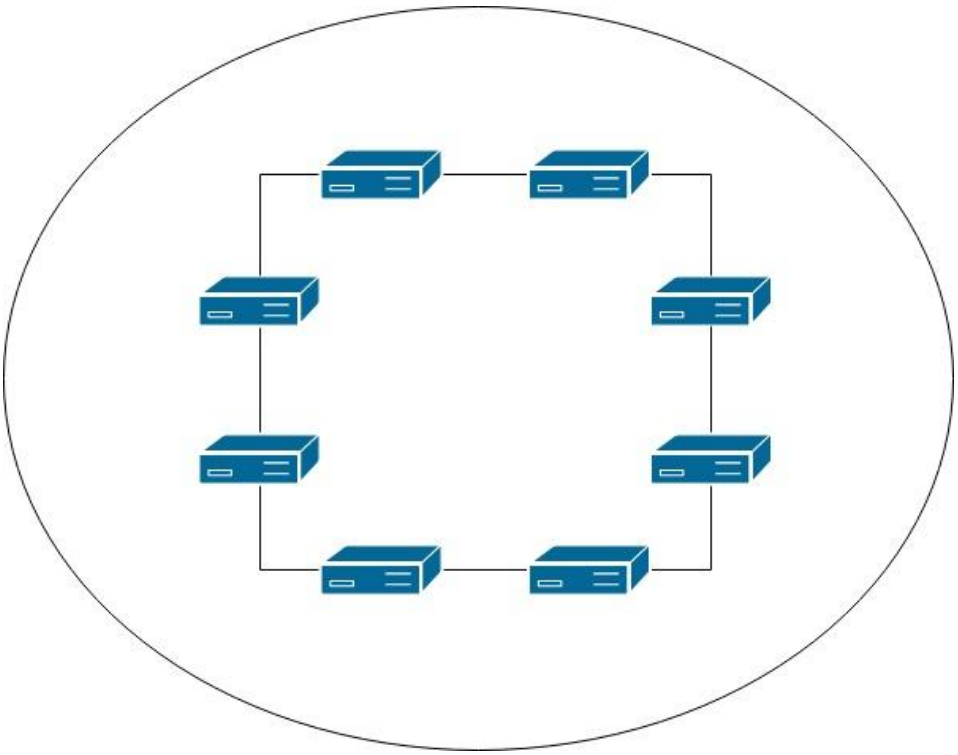


Рисунок 3.2 - Децентралізована глобальна структура рівня управління

Ієрархічна структура, як правило, має 2 рівня. В даній роботі розглядається ієрархічна структура з двома рівнями, оскільки ієрархічна структура з більшою кількістю рівнів може бути розглянута в такий же самий спосіб. Схематичне представлення ієрархічної структури можна побачити на рисунку 3.3.

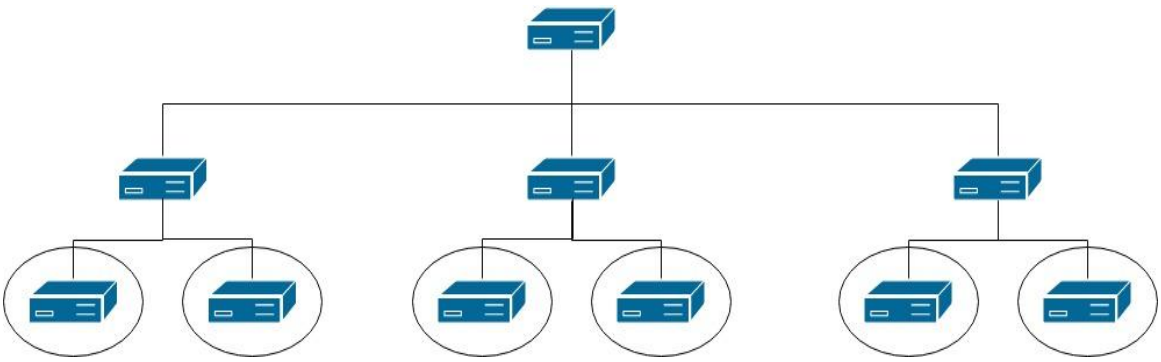


Рисунок 3.3 - Ієрархічна структура рівня управління

В даній структурі рівень управління організовано у вигляді дерева, і вузли розділені на два типи: один корінний вузол і багато листів. Вузли-листя

управляють безпосередньо рівнем даних. І видимість кожного вузла-листа поширюється тільки на певну зону в мережі і управляється коренем дерева. Контролер кожного окремого листа і його локальна мережа абстрагуються коренем, як логічний вузол. Корінь володіє інформацією про ці логічні вузли. Потоки можуть бути двох видів: локальними і глобальними. Початковий хост і хост призначення управляються одним і тим же листовим контролером. Глобальні потоки повністю навпаки. Кожен запит ініціалізації локального потоку буде оброблений тільки одним листовим контролером. В той час як кожен запит ініціалізації глобального потоку буде оброблений кореневим контролером спочатку, а потім розділений на локальні запити, які будуть оброблятися відповідними листовими контролерами.

Припустимо існує m_H листових контролерів і середня дистанція листових контролерів d_H. З цього випливає, що кореневий контролер управляє логічним графом з m_H вузлами. Таким чином, середній час обробки запита кореневим контролером буде:

$$\mu_{H,r} = \frac{K}{g(m_H)}.$$

Кожен листовий контролер управляє $\frac{N}{m_H}$ хостами. З цього вираховуємо середній час обробки запиту для кожного листового контролера:

$$\mu_{H,l} = \frac{K}{g(\frac{N}{m_H})}.$$

Будемо позначати *j*-ий контролер *i*-го рівня як $c_{i,j}$. $C_{x,y}$ - позначення множини, до якої входять контролери, які необхідно залучити, щоб обробити запити ініціації потоку $f_{x,y}$. Нехай $I_{x,y}(i,j)$ буде бінарною змінною, що ідентифікує, чи входить $c_{i,j}$ до $C_{x,y}$.

$$I_{x,y}(i,j) = \begin{cases} 1; якщо\ c_{i,j} \in C_{x,y} \\ 0; \end{cases}$$

Серед N^2 - N потоків $\frac{N^2}{m_H} - N$ локальних потоків і $N^2 - \frac{N^2}{m_H}$ глобальних. Таким чином, можна отримати:

$$\sum_{x=1}^{N}\sum_{y=1,y\neq x}^{N} I_{x,y}(1,1) = N^2 - \frac{N^2}{m_H}$$

$$\sum_{x=1}^{N}\sum_{y=1,y\neq x}^{N}\sum_{j=1}^{m_H} I_{x,y}(2,j) = (N^2 - \frac{N^2}{m_H}) \times (d_H + 1) + \frac{N^2}{m_H} - N$$

Отже, середня кількість запитів ініціації потоку в кореневому контролері визначається як:

$$\lambda_{H,r} = \lambda \times \sum_{x=1}^{N}\sum_{y=1,y\neq x}^{N} I_{x,y}(1,1) = \lambda \times (N^2 - \frac{N^2}{m_H})$$

І середній час прибуття запиту до кожного листового контролера буде:

$$\lambda_{H,l} = \lambda \times \sum_{x=1}^{N}\sum_{y=1,y\neq x}^{N}\sum_{j=1}^{m_H}\frac{I_{x,y}(2,j)}{m_H} = \lambda \times ((N^2 - \frac{N^2}{m_H}) \times (d_H + 1) + \frac{N^2}{m_H} - N).$$

Нехай T_H - час відгуку на запит ініціалізації потоку. T_H можна розділити на дві частини: час відгуку *Tr* на кореневому контролері і час відгуку *Tl* на листовому контролері. Отримуємо $E\{T_H\} = E\{Tr\} + E\{Tl\}$. Нехай $Tr_{x,y}$ буде час відгуку на запит ініціації потоку $f_{x,y}$ на кореневому контролері (якщо $f_{x,y}$ це локальний потік, то $Tr_{x,y} = 0$). $Tc_{i,,j}$ - час відгуку для запиту ініціації потоку $c_{i,,j}$.

Тож отримуємо:

$$E\{Tr\} = \frac{\sum_{x=1}^{N}\sum_{y=1,y\neq x}^{N} E\{Tr_{x,y}\}}{N \times (N-1)} = \frac{\sum_{x=1}^{N}\sum_{y=1,y\neq x}^{N} E\{Tc_{1,1}\} \times I_{x,y}(1,1)}{N \times (N-1)}$$

$$= \frac{N - \frac{N}{m_H}}{N-1} \times \frac{1}{\mu_{H,r} - \lambda_{H,r}}$$

Якщо $f_{x,y}$ це глобальний потік, запит ініціації генерований $f_{x,y}$ буде поділений кореневим контролером на $\sum_{j=1}^{m_H} I_{x,y}(2,j)$ локальних запитів. Таким чином, $Tl_{x,y}$ буде дорівнювати найдовшому часу відгуку для локальних запитів. Тоді:

$$Tl_{x,y} = Max_{j=1}^{m_H} I_{x,y}(2,j) \times Tc_{2,j}$$

Якщо $f_{x,y}$ - це локальний потік, вираз $Tl_{x,y}$ залишиться без змін. Оскільки $Tc_{2,j}(j = 1,2, \dots, m_H)$ є незалежною однаково розподіленою величиною і має негативне експоненціальне розподілення з середнім значенням $\frac{1}{\mu_{H,l}-\lambda_{H,l}}$, то

$$P\{Tl_{x,y} < t\} =$$

$$\prod_{j=1}^{m_j} P\{I_{x,y}(2,j) \times Tc_{2,j} < t\} = (1 - e^{(\lambda_{H,r}-\mu_{H,r})\times t})^{\sum_{j=1}^{m_H} I_{x,y}(2,j)}.$$

Нехай $d_x = \sum_{j=1}^{m_H} I_{x,y}(2,j)$. Тоді функція щільності ймовірності буде

$$f_{Tl_{x,y}}(t) = d_{x,y} \times (1 - e^{(\lambda_{H,r}-\mu_{H,r})\times t})^{d_{x,y}-1} \times (\lambda_{H,r} - \mu_{H,r}) \times$$

$$e^{(\lambda_{H,r}-\mu_{H,r})\times t}.$$

Таким чином, середнє значення $Tl_{x,y}$ буде

$$E\{Tl_{x,y}\} = \int_0^{\infty} f_{Tl_{x,y}}(t) \times t\, dt$$

$$= \frac{d_{x,y}}{\lambda_{H,r} - \mu_{H,r}} \times \sum_{i=0}^{d_{x,y}-1} \binom{d_{x,y} - 1}{i} \times \frac{(-1)^i}{d_{x,y}^2} \leq \frac{\ln d_{x,y} + 1}{\lambda_{H,r} - \mu_{H,r}}.$$

Виходячи з цього, час середньої відповіді контролера в ієрархічній структурі буде

$$E\{T_H\} = \frac{\frac{N-\frac{N}{m_H}}{N-1}}{\mu_{H,r}-\lambda_{H,r}} + \frac{\ln\left(\frac{N-\frac{N}{m_H}}{N-1}\times d_H+1\right)+1}{\mu_{H,l}-\lambda_{H,l}}.$$

При розрахунках середнього часу відгуку контролера в різних структурах були отримані результати, наведені в таблиці 3.1. Для розрахунку бралися наступні дані: кількість вузлів N = 100, швидкість прибуття запитів в секунду λ = 10, кількість контролерів в децентралізованих структурах m_D = 10 і середня дистанція контролерів d_D = 5.

Таблиця 3.1 - Середній час відгуку контролера в різних структурах

Тип структури	Централізована	Децентралізована (локальна)	Децентралізована (глобальна)	Ієрархічна
E, c	$1{,}194 \times 10^{-4}$	$1{,}9 \times 10^{-6}$	$1{,}026 \times 10^{-5}$	$3{,}91 \times 10^{-6}$

Базуючись на результатах досліджень, можна зробити висновок, що найбільш ефективною топологією рівня управління є децентралізована з локальною видимістю мережі. Проте, в даному випадку ми розглядали мережу, де один контролер відповідає за свою підмережу. Це виглядає як об'єднання централізованих мереж. За результатами підрахунків, мережа централізована архітектура рівня контролю найгірше масштабується і має найгірші показники середнього часу відгуку контролера. Окрім того, навантаження на один контролер може бути значним. Тобто, не дивлячись на те, що при децентралізованій локальній архітектурі, кожному контролеру потрібно оброблювати запити зі значно меншої кількості вузлів, ніж в випадку з децентралізованою локальною структурою, все одно можливі випадки, коли навантаження на один контролер в підмережі буде значно більше очікуваного. А це негативно впливає на масштабованість рівня управління і на надійність мережі в цілому, оскільки, при відмовленні одного єдиного контролера, що управляє підмережею, можуть виникнути проблеми.

Значно кращим варіантом буде використання декількох контролерів, замість одного, для управління підмережею. Фактично децентралізувати контроль підмережі. Використовувати для даних цілей децентралізовану локальну або ієрархічну архітектуру занадто складно. Оскільки, в будь-якому випадку організація цих топологій потребує більше ресурсів і часу для налаштування, ніж звичайна глобальна децентралізована архітектура. Окрім того, якщо вузлів в підмережі не так багато, ніякої різниці зі звичною глобальною децентралізованою топологією не буде.

Тому було вирішено поєднати децентралізовану локальну і децентралізовану глобальну структури. На рисунку 3.4 можна побачити схематичне зображення цієї архітектури.

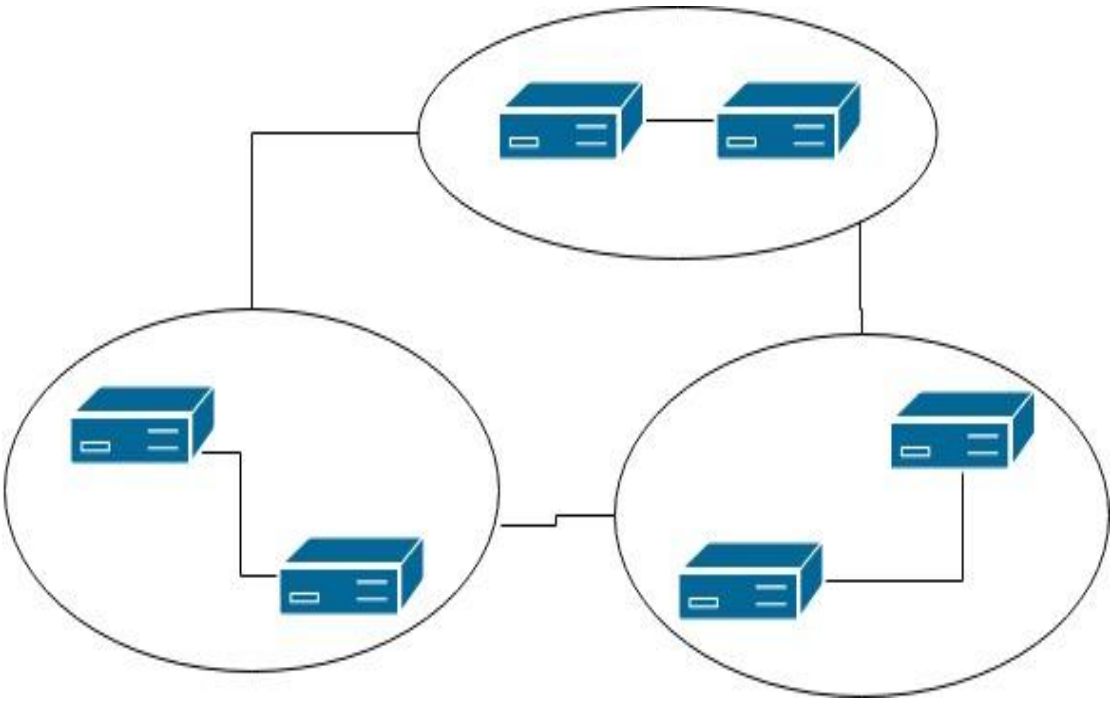


Рисунок 3.4 - Децентралізована гібридна архітектура (локально-глобальна)

$$\lambda_{sub} = \frac{\frac{N}{M} \times (\frac{N}{M} - 1) \times \lambda}{m_D},$$

$$\mu_{sub} = \frac{K}{g(\frac{N}{M})}.$$

$$E\{T_C(N)\} = \frac{\frac{N}{M}}{\mu_{sub} - \lambda_{sub}}.$$

Будемо вважати групу контролерів єдиним цілим. Тому надалі необхідно використовувати формули для розрахунків показників децентралізованої локальної архітектури. Замість кількості фізичних вузлів використовується кількість угрупувань M.

$$\lambda_{main} = \lambda \times \frac{(M^2 - \frac{M^2}{m_D}) \times (d_D + 1) + (\frac{N^2}{m_D} - M)}{m_D},$$

$$\mu_{main} = \frac{K}{g(\frac{M}{m_D} + m_D - 1)},$$

Звідси середній час відгуку контролера:

$$E\{T_{D,l}\} = \frac{1 + \frac{M \times (m_D - 1)}{(M-1) \times m_D} \times d_D}{\mu_{main} - \lambda_{main}} + \frac{\frac{N}{M}}{\mu_{sub} - \lambda_{sub}}$$

3.3 Аналіз отриманого результату

Як можна побачити в таблиці 3.2, при використанні гібридної децентралізованої структури можна отримати майже в 5 разів менший час відгуку контролера.

Таблиця 3.2 - Середній час відгуку контролера в різних структурах з урахуванням гібридної

Тип структури	Централізована	Децентралізована (локальна)	Децентралізована (глобальна)	Ієрархічна	Децентралізована (гібридна)
$E,\ c$	$1{,}194 \times 10^{-4}$	$1{,}9 \times 10^{-6}$	$1{,}026 \times 10^{-5}$	$3{,}91 \times 10^{-6}$	$4{,}7 \times 10^{-7}$

Висновки до розділу 3

Проведено аналіз існуючих структур архітектури рівня контролю. Математично визначена середній час відгуку контролера.

Запропоновано модифікований спосіб структуризації рівня управління SDN, який відрізняється від відомих більшою швидкодією приблизно в 5 разів.

ВИСНОВКИ

Проведено аналіз основних принципів побудови SDN. Проаналізовано передумови виникнення SDN, з яких можна зробити висновок, що концепція програмно-конфігуровних мереж має великий потенціал і допомагає вирішити проблеми, що зустрічаються в класичних підходах до організації мереж, зокрема зменшити витрати на організацію мережі за рахунок використання більш простих пристроїв і спростити контроль за мережею, за рахунок чіткого відокремлення рівня даних і рівня управління.

Проаналізовані особливості загальної архітектури мережі, що побудована за даною концепцією і ключові деталі. Досліджені основні протоколи для організації цього виду мереж, а також основні типи додатків.

Проведено детальний аналіз різних типів архітектури рівня управління. Проаналізована роль кожного елемента, що входить до рівня управління. Освітлено і проаналізовано найпоширеніші проблеми рівня управління.

Показано, що на сьогодні хоча і розроблені декілька видів архітектур рівня управління, та залишається невирішеним питання масштабованості і швидкодії мережі. Крім того, необхідно приділяти увагу безпеці рівня контролю.

Проведено аналіз існуючих структур архітектури рівня контролю. Математично визначена середній час відгуку контролера. Запропоновано власний спосіб структуризації рівня управління SDN, який відрізняється від відомих більшою швидкодією приблизно в 5 разів.

Література

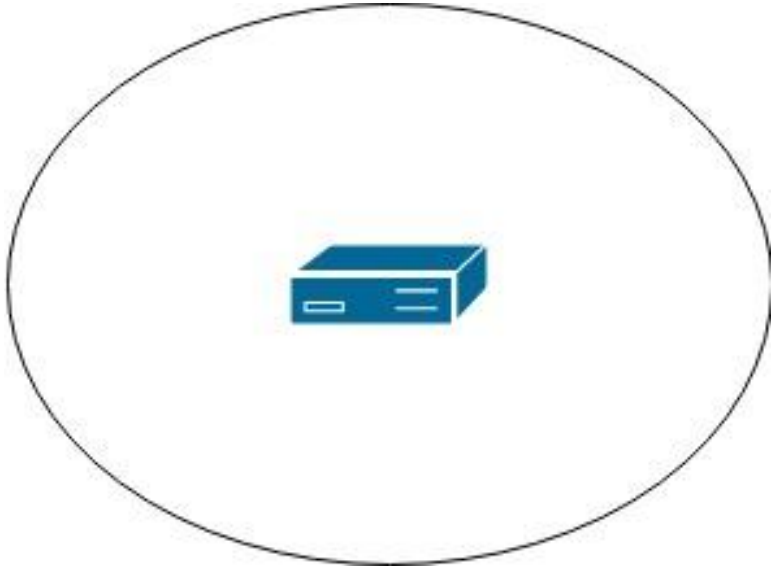
1. SDN basics: Understanding centralized control and programmability [Електронний ресурс] / / TechTarget. - Режим доступу: https://searchsdn.techtarget.com/tip/SDN-basics-Understanding-centralized-control-and-programmability, вільний. - Загл. з екрану. (20.03.2018).
2. *Nadeau T.* SDN – Software Defined Networks. / T. Nadeau — CA.: published by O'reilly media, 2013. – 80 с.
3. *Goransson P.* Software Defined Networks: A Comprehensive Approach. / P. Goransson. — MA.: Elsevier Inc, 2014. — С. 215.
4. *Azodolmolky S.* Software Defined Networking with OpenFlow / S. Azodolmolky – UK.: Packt Publishing, 2013. – 148 с.
5. Software-Defined Networking (SDN) Definition [Електронний ресурс] / / ONF. - Режим доступу: https://www.opennetworking.org/sdn-definition/, вільний. - Загл. з екрану. (28.03.2018).
6. Критерії оцінки мереж [Електронний ресурс] / / om.net.ua. - Режим доступу: http://om.net.ua/9/9_6/9_68601_kriterii-otsenki-setey.html, вільний. - Загл. з екрану. (03.04.2018).
7. *Monge A.* MPLS in the SDN Era. / *Monge A.* — CA.: published by Juniper networkds, 2013. – 194 с.
8. *Stallings W.* Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud / *Stallings W.* — CA.: published by Pearson, 2015. – 116 с.
9. *Doherty J.* SDN and NFV Simplified / *Doherty J.* — CA.: published by Addison-Wesley Professional, 2016. – 173 с.
10. *Tiwari V.* SDN and OpenFlow for Beginners with Hands on Labs / *Tiwari V.* — CA.: published by Amazon Digital Services LLC, 2013. – 15 с.

Додаток 1
Копії графічних матеріалів

Визначення основних показників в централізованій структурі

Швидкість надходження запитів на контролер

$$\lambda_C = N \times (N-1) \times \lambda$$



Середній час обробки запиту

$$\mu_C = \frac{K}{g(N)}$$

Середній час відгуку контролера

$$E\{T_C(N)\} = \frac{1}{\mu_C - \lambda_C}$$

Визначення основних показників в децентралізованій локальній структурі

Швидкість надходження запитів на контролер

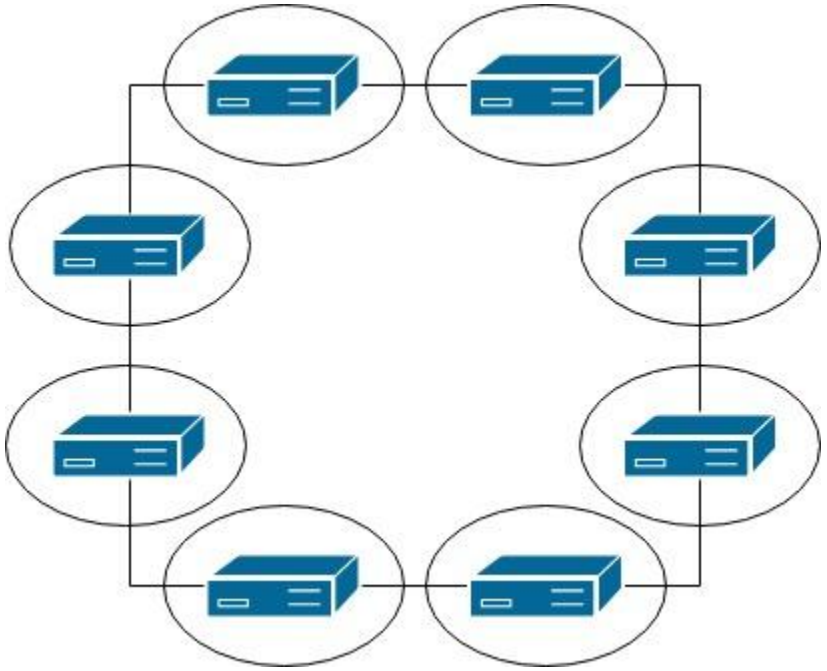
$$\lambda_{D,l} = \lambda \times \frac{(N^2 - \frac{N^2}{m_D}) \times (d_D + 1) + (\frac{N^2}{m_D} - N)}{m_D},$$

Середній час обробки запиту

$$\mu_{D,l} = \frac{K}{g(\frac{N}{m_D} + m_D - 1)},$$

Середній час відгуку контролера

$$E\{T_{D,l}\} = \frac{1 + \frac{N \times (m_D - 1)}{(N-1) \times m_D} \times d_D}{\mu_{D,l} - \lambda_{D,l}}.$$



Визначення основних показників в децентралізованій глобальній структурі

Швидкість надходження запитів на контролер

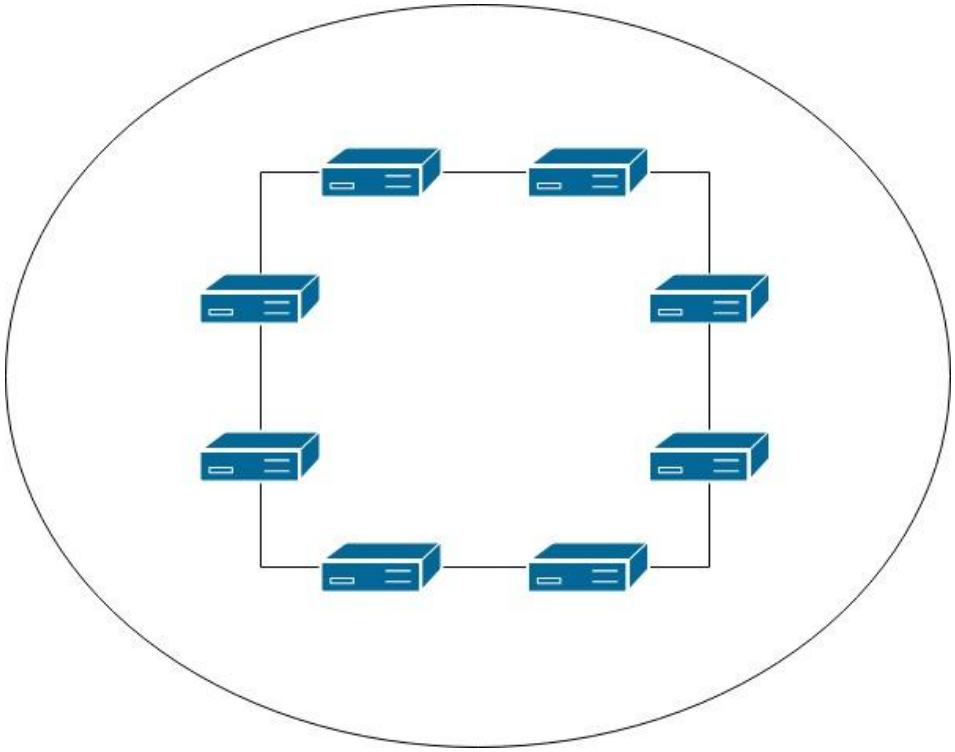
$$\lambda_{D,g} = \frac{N \times (N-1) \times \lambda}{m_D}.$$

Середній час обробки запиту

$$\mu_{D,g} = \frac{K}{g(N)}$$

Середній час відгуку контролера

$$E\{T_{D,g}\} = \frac{1}{\mu_C - \lambda_C}$$



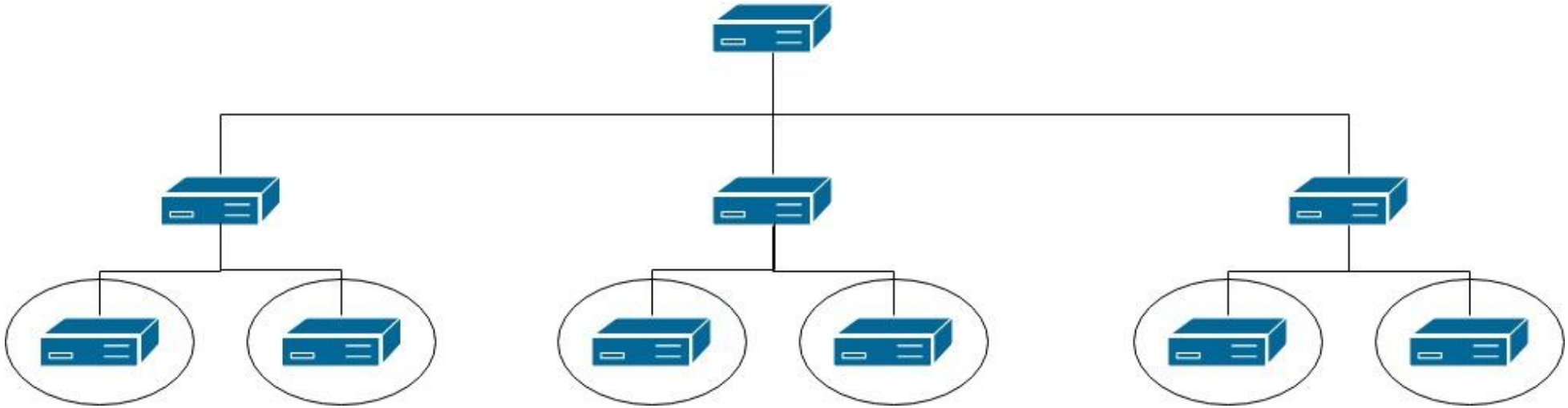
Визначення основних показників в ієрархічній структурі

Швидкість надходження запитів на контролер

$$\lambda_{H,r} = \lambda \times (N^2 - \frac{N^2}{m_H}) \qquad \lambda_{H,l} = \lambda \times ((N^2 - \frac{N^2}{m_H}) \times (d_H + 1) + \frac{N^2}{m_H} - N)$$

Середній час обробки запиту Середній час відгуку контролера

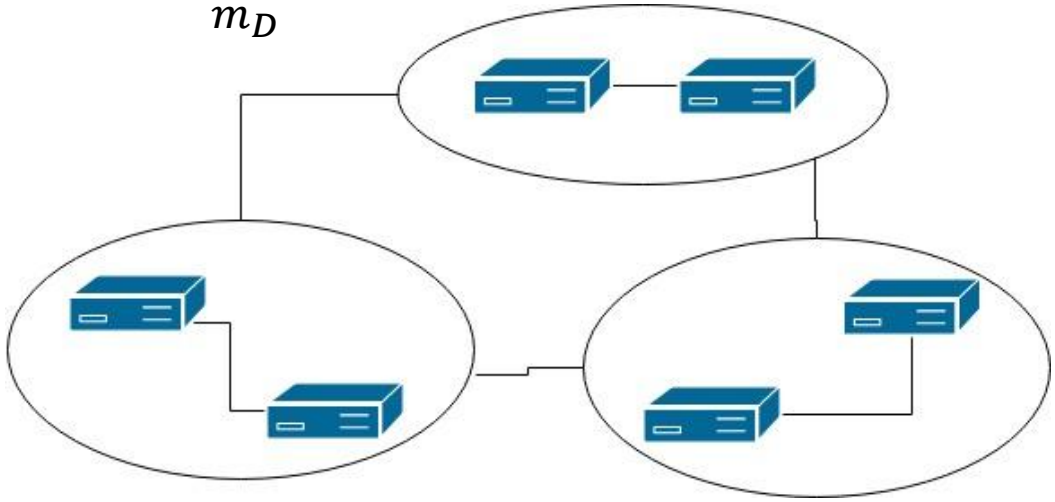
$$\mu_{H,r} = \frac{K}{g(m_H)} \qquad \mu_{H,l} = \frac{K}{g(\frac{N}{m_H})} \qquad E\{T_C(N)\} = \frac{1}{\mu_C - \lambda_C}$$



Визначення основних показників в модифікованій структурі

Швидкість надходження запитів на контролер

$$\lambda_{sub} = \frac{\frac{N}{M}\times(\frac{N}{M}-1)\times\lambda}{m_D} \qquad \lambda_{main} = \lambda \times \frac{(M^2-\frac{M^2}{m_D})\times(d_D+1)+(\frac{N^2}{m_D}-M)}{m_D}$$



Середній час обробки запиту

$$\mu_{sub} = \frac{K}{g(\frac{N}{M})} \qquad \mu_{main} = \frac{K}{g(\frac{M}{m_D}+m_D-1)}$$

Середній час відгуку контролера

$$E\{T_{D,l}\} = \frac{1+\frac{M\ \times(m_D-1)}{(M-1)\times m_D}\times d_D}{\mu_{main}-\lambda_{main}} + \frac{\frac{N}{M}}{\mu_{sub}-\lambda_{sub}}$$

Час відгуку контролера в різних структурах рівня управління

Тип структури	Централізована	Децентралізована (локальна)	Децентралізована (глобальна)	Ієрархічна	Децентралізована (гібридна)
E, c	$1{,}194 \times 10^{-4}$	$1{,}9 \times 10^{-6}$	$1{,}026 \times 10^{-5}$	$3{,}91 \times 10^{-6}$	$4{,}7 \times 10^{-7}$

Додаток 2
Копії публікацій

ISSN 2520-2057

МІЖНАРОДНИЙ НАУКОВИЙ ЖУРНАЛ «ІНТЕРНАУКА»

INTERNATIONAL SCIETIFIC JOURNAL «INTERNAUKA»

МЕЖДУНАРОДНЫЙ НАУЧНЫЙ ЖУРНАЛ «ИНТЕРНАУКА»

№ 7 (47) / 2018
2 том

Технічні науки

УДК 044.77

Орлова Марія Миколаївна

кандидат технічних наук, доцент кафедри

системного програмування і спеціалізованих комп'ютерних систем

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Орлова Мария Николаевна

кандидат технических наук, доцент кафедры

системного программирования и специализированных компьютерных стистем

Национальный технический университет Украины

«Киевский политехнический институт имени Игоря Сикорского»

Mariia Orlova

PhD, Assistant Professor of Department of

System Programming and Specialized Computer Systems

National Technical University of Ukraine

"Igor Sikorsky Kyiv Polytechnic Institute"

Єрмоленко Ігор Андрійович

магістрант

Національного технічного університету України

«Київський політехнічний інститут імені Ігоря Сікорського»

Ермоленко Игорь Андреевич

магистрант

Национального технического университета Украины

«Киевский политехнический институт имени Игоря Сикорского»

Ihor Yermolenko

Student of the

National Technical University of Ukraine

«Igor Sikorsky Kyiv Polytechnic Institute»

МАТЕМАТИЧНЕ ПОРІВНЯННЯ ПРОДУКТИВНОСТІ ТОПОЛОГІЙ РІВНЯ УПРАВЛІННЯ ПРОГРАМНО-КОНФІГУРОВНИХ МЕРЕЖ

МАТЕМАТИЧЕСКОЕ СРАВНЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ ТОПОЛОГИЙ УРОВНЯ УПРАВЛЕНИЯ ПРОГРАММНО-КОНФИГУРИРУЕМЫХ СЕТЕЙ

MATH COMPARSION OF PERFORMANCE OF SOFTWARE DEFINED NETWORKING CONTROL PLANE TOPOLOGIES

Анотація. *Проведений аналіз масштабованості рівня управління SDN, в яких вузли мають подібний обіг трафіку. Проаналізовані стандартні структури рівня управління, а саме - централізована, локально децентралізована і глобально децентралізована. Запропонована модифікована топологія рівня управління.*

Ключові слова: *SDN, масштабованість мереж, рівень управління, час відгуку контролера, OpenFlow.*

Аннотация. *Проведен анализ масштабируемости топологий уровня управления SDN, в которых узлы имеют подобное обращение трафика. Проанализированы стандартные структуры уровня управления, а именно централизованная, локально децентрализованная и глобально децентрализованная. Была предложена модифицированная топология уровня управления.*

Ключевые слова: *SDN, масштабируемость сетей, уровень контроля, время отклика контроллера, OpenFlow.*

Summary. *Scalability of the topologies of the SDN control level where nodes have a similar traffic flow was analyzed. The standard control structures as centralized, locally decentralized and globally decentralized were analyzed. Modified topology of control plane was proposed*

Key words: *SDN, networks scalability, control plane, controller response time, OpenFlow.*

Існують два загальноприйнятих варіанти поліпшення продуктивності рівня управління SDN. Перший – це перенести частину функції управління на інші пристрої в мережі, наприклад, мережеві комутатори. Цей підхід дійсно дозволяє значно зменшити навантаження на центральний вузол, і значно збільшує загальну продуктивність мережі. Проте, комутатори повинні бути побудовані на спеціальній інтегральній схемі і мати центральний процесор. Таким чином, збільшується складність розгортання мережі. Другий підхід полягає в проектуванні особливого розподіленого рівня управління, який передбачає розподілення навантаження між декількома контролерами. Іншими словами, рівень управління буде працювати як розподілена комп'ютерна система, що колективно обробляє вхідні мережеві запити.

Як правило, архітектура рівня управління SDN може бути централізованою і децентралізованою. Децентралізована, в свою чергу, поділяється на два підвиди: локальний і глобальний. Вузли локального виду не бачать мережу в цілому. Кожен вузол обробляє запити тільки від частини вузлів мережі. В свою чергу, вузли в децентралізованих системах глобального виду можуть приймати запити від усіх вузлів мережі. Децентралізовані структури також називають розподіленими структурами [1].

Метою роботи є математичне порівняння різних типів архітектури рівня управління SDN та створення власної структури рівня управління.

Навантажувальна спроможність контролера визначається центральним процесором, використанням пам'яті і завантаженістю мережі. Як правило, "вузьким місцем" є процесор. Тому будемо приймати до уваги тільки час, який витрачає центральний процесор для обробки одного запиту. Час обробки залежить від топології мережі, використаних алгоритмів маршрутизації і обчислювальної потужності контролера. Часова складність

алгоритмів маршрутизації позначається $g(V,E)$, де V – кількість мережевих вузлів, а E – кількість мережевих з'єднань. Тоді ми припустимо, що час обробки підлягає експоненційному розподілу з середнім значенням $(g(V,E))/K$, де K – обчислювальна потужність [2]. Тому припустимо, що контролер використовує алгоритм Дейкстри для пошуку найкращого шляху. Складність алгоритму Дейкстри залежить від реалізації, і у загальному випадку складає $O(V^2)$. Для спрощення обчислень, в цій роботі допустимо, що $g(V,E) = V^2$. Оскільки надходження запитів з одного вузла до іншого піддається Пуассонівському розподіленню, а сума незалежних випадкових величин Пуассонівського розподілення представляє собою Пуассонівське розподілення, то сукупність запитів, що надходять до контролера, також піддаються Пуассонівському розподіленню. Таким чином, кожен контролер є моделлю черги *M/M/1* [3].

Базуючись на результатах досліджень, можна зробити висновок, що найбільш ефективною топологією рівня управління є децентралізована з локальною видимістю мережі. Проте, в даному випадку ми розглядали мережу, де один контролер відповідає за свою підмережу. Це виглядає як об'єднання централізованих мереж. За результатами підрахунків, мережа централізована архітектура рівня контролю найгірше масштабується і має найгірші показники середнього часу відгуку контролера. Окрім того, навантаження на один контролер може бути значним. Тобто, не дивлячись на те, що при децентралізованій локальній архітектурі, кожному контролеру потрібно оброблювати запити зі значно меншої кількості вузлів, ніж в випадку з децентралізованою локальною структурою, все одно можливі випадки, коли навантаження на один контролер в підмережі буде значно більше очікуваного. А це негативно впливає на масштабованість рівня управління і на надійність мережі в цілому, оскільки, при відмовленні одного єдиного контролера, що управляє підмережею, можуть виникнути проблеми.

Значно кращим варіантом буде використання декількох контролерів, замість одного, для управління підмережею. Фактично децентралізувати

контроль підмережі. Використовувати для даних цілей децентралізовану локальну архітектуру занадто складно. Оскільки, в будь-якому випадку організація цих топологій потребує більше ресурсів і часу для налаштування, ніж звичайна глобальна децентралізована архітектура. Окрім того, якщо вузлів в підмережі не так багато, ніякої різниці зі звичною глобальною децентралізованою топологією не буде.

Тому було вирішено поєднати децентралізовану локальну і децентралізовану глобальну структури. На рисунку 1 представлене схематичне зображення цієї архітектури.

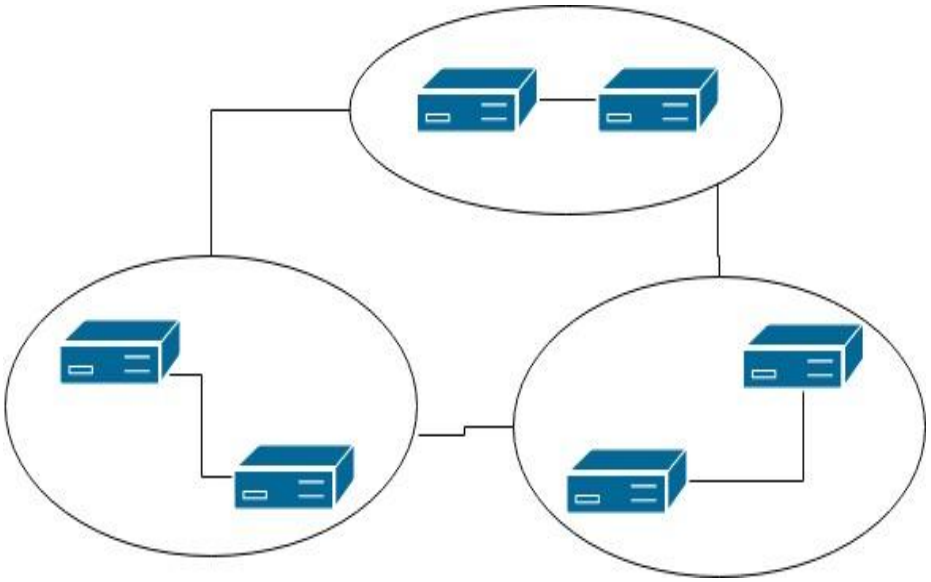


Рисунок 1 - Гібридна топологія рівня контролю

Спочатку необхідно розрахувати швидкодію для об'єднань контролерів, тобто для децентралізованих глобальних контролерів. Тому для цих цілей будуть використовуватися ті ж самі формули, що і для глобальних контролерів. Різниця буде в кількості вузлів. Припустимо, що вони рівномірно розташовані відносно контролерів. Тоді, для кожної сукупності контролерів будемо використовувати N/M, де M - кількість груп контролерів.

Будемо вважати групу контролерів єдиним цілим. Тому надалі необхідно використовувати формули для розрахунків показників децентралізованої локальної архітектури. Замість кількості фізичних вузлів використовується кількість угрупувань M.

При розрахунках середнього часу відгуку контролера в різних структурах були отримані результати, наведені в таблиці 1. Для розрахунку бралися наступні дані: кількість вузлів $N = 100$, швидкість прибуття запитів в секунду $\lambda = 10$, кількість контролерів в децентралізованих структурах $m_D = 10$ і середня дистанція контролерів $d_D = 5$.

Таблиця 1

Час відгуку контролера в різних архітектурах з заданими параметрами

Тип структури	Централізована	Децентралізована (локальна)	Децентралізована (глобальна)	Децентралізована (гібридна)
$E,\ c$	$1{,}194 \times 10^{-4}$	$1{,}9 \times 10^{-6}$	$1{,}026 \times 10^{-5}$	$4{,}7 \times 10^{-7}$

Висновки. В результаті проведених досліджень показано, що гібридна децентралізована архітектура рівня управління зменшує час відгуку контролера приблизно в 4 рази, порівняно зі стандартними архітектурами рівня управління. А це означає, що модифікована архітектура рівня контролю дозволяє масштабуватися мережі з найменшою втратою швидкодії порівняно з іншими архітектурами.

Література

1. Nadeau T. SDN – Software Defined Networks. / T. Nadeau — CA.: published by O'reilly media, 2013. – 80 с.
2. Goransson P. Software Defined Networks: A Comprehensive Approach. / P. Goransson. — MA.: Elsevier Inc, 2014. — С. 215.
3. Azodolmolky S. Software Defined Networking with OpenFlow / S. Azodolmolky – UK.: Packt Publishing, 2013. – 148 с.

Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»
Факультет прикладної математики

1898

ПРИКЛАДНА МАТЕМАТИКА ТА КОМП'ЮТИНГ ПМК 2018

Десята наукова конференція
магістрантів та аспірантів

Київ, 21–23 березня 2018 р.

Збірник тез доповідей

ЗМІСТ

УДК 004.77

К.т.н., доцент Орлова М.М., студент Єрмоленко І.А.

**Національний технічний університету України
«Київський політехнічний інститут імені Ігоря Сікорського»**

ПОРІВНЯННЯ МАСШТАБОВАНОСТІ РІЗНИХ ТИПІВ РІВНЯ УПРАВЛІННЯ В ПРОГРАМНО-КОНФІГУРОВАНИХ МЕРЕЖАХ

Abstract

Maria Orlova, assoc. prof., PhD; Ihor Yermolenko, student

Scalability comparison of different types of control levels in program-defined networks

One of the most important parameters of network is scalability. The more nodes appear in network, the more load comes to controllers. And software-defined networks are not an exception.

Вступ

З розвитком Інтернету контроль за мережею став вузьким місцем в класичному мережевому підході. Це спричинило появу нової програмно-конфігурованої архітектури SDN (Software defined network), в якій рівень управління мережею відділений від пристроїв передачі даних і реалізується програмно. SDN розділяє рівень контролю і рівень даних, що призводить до значного спрощення модуля управління. Таким чином вся логіка управління мережею зосереджена, наприклад, на одному сервері, а інші пристрої мережі, відповідно до концепції, використовуються тільки для передачі даних. Проте одним із основних недоліків нової архітектури є складність масштабування рівня контролю за мережею. Зі збільшенням кількості вузлів у мережі, і, відповідно, зі значним збільшенням запитів до рівня управління, навантаження на модуль контролю зростає, і його продуктивність може значною мірою знижуватися. Таким чином, обчислення масштабованості рівня управління SDN є критичною проблемою для успішної адаптації SDN до великомасштабних мереж або мереж з великою кількістю потоків.

Постановка задачі

Оскільки мережі SDN використовують новий принцип побудови мереж, переважна більшість існуючих досліджень не створюють модель, а базуються на експериментах і моделюванні завдяки спеціалізованому програмному забезпеченню. Метою роботи є математичне порівняння різних типів архітектури рівня управління SDN.

Математичне порівняння різних типів архітектури модуля контролю

Існують два загальноприйнятих варіанти поліпшення продуктивності рівня управління SDN. Перший – це перенести частину функції управління на інші пристрої в мережі, наприклад, мережеві комутатори. Цей підхід дійсно дозволяє значно зменшити навантаження на центральний вузол, і значно збільшує загальну продуктивність мережі. Проте, комутатори повинні бути побудовані на спеціальній інтегральній схемі і мати центральний процесор. Таким чином, збільшується складність розгортання мережі. Другий підхід полягає в проектуванні особливого розподіленого рівня управління, який передбачає розподілення навантаження між декількома контролерами. Іншими словами, рівень управління буде працювати як розподілена комп'ютерна система, що колективно обробляє вхідні мережеві запити.

Як правило, архітектура рівня управління SDN може бути централізованою і децентралізованою. Децентралізована, в свою чергу, поділяється на два підвиди: локальний і глобальний. Вузли локального виду не бачать мережу в цілому. Кожен вузол обробляє запити тільки від частини вузлів мережі. В свою чергу, вузли в децентралізованих системах глобального виду можуть приймати запити від усіх вузлів мережі. Децентралізовані структури також називають розподіленими структурами.

Для розрахунків масштабованості рівня управління SDN можна використовувати метрики масштабованості для розподіленої системи. Масштабованість для розподілених систем базується на продуктивності. Розподілену систему можна назвати масштабованою, якщо продуктивність системи залишається на одному рівні при збільшенні розміру системи. Для рівня управління SDN продуктивність $F(N)$ може бути визначена як:

$$F(N) = \phi(N) \times \frac{T(N)}{C(N)},$$

де, N – кількість вузлів в мережі, $\phi(N)$ – пропускна спроможність рівня контролю в обробці мережевих запитів, $T(N)$ – середній час відгуку на кожен запит, $C(N)$ – вартість розгортання рівня управління (наприклад, вартість одного контролеру).

Таким чином, масштабованість для рівня управління SDN, розмір якої змінюється від N_2 до N_1 визначається як:

$$\Psi(N_1, N_2) = \frac{F(N_2)}{F(N_1)}.$$

В SDN існує декілька типів мережевих запитів, які рівень управління повинен обробляти. Наприклад, запит для спостерігання за станом мережі, оновлення стану мережі, відновлення після збою. Однак ініціювання потоку є головною і основною функціональністю контролеру SDN. Тому основна увага приділяється обробці ініціювання потоку в рівні управління. В

децентралізованих структурах локального вигляду та ієрархічних структурах обробка запиту ініціювання потоку може оброблятися декількома контролерами. Необхідно ввести такий структурний параметр, як середня дистанція контролерів, що визначає яка кількість контролерів в середньому необхідна для обробки запиту ініціювання потоку.

Навантажувальна спроможність контролера визначається центральним процесором, використанням пам'яті і завантаженістю мережі. Як правило, "вузьким місцем" є процесор. Тому будемо приймати до уваги тільки час, який витрачає центральний процесор для обробки одного запиту. Час обробки залежить від топології мережі, використаних алгоритмів маршрутизації і обчислювальної потужності контролера. Часова складність алгоритмів маршрутизації позначається $g(V,E)$, де V – кількість мережевих вузлів, а E – кількість мережевих з'єднань. Тоді ми припустимо, що час обробки підлягає експоненційному розподілу з середнім значенням $\frac{g(V,E)}{K}$, де K – обчислювальна потужність. Тому припустимо, що контролер використовує алгоритм Дейкстри для пошуку найкращого шляху. Складність алгоритму Дейкстри залежить від реалізації, і у загальному випадку складає $O(V^2)$. Для спрощення обчислень, в цій роботі допустимо, що $g(V,E) = V^2$. Оскільки надходження запитів з одного вузла до іншого піддається Пуассонівському розподіленню, а сума незалежних випадкових величин Пуассонівського розподілення представляє собою Пуассонівське розподілення, то сукупність запитів, що надходять до контролера, також піддаються Пуассонівському розподіленню. Таким чином, кожен контролер є моделлю черги M/M/1.

Спочатку визначимо час відгуку контролера в централізованій структурі при кількості вузлів N та базовій середній інтенсивності надходження запитів λ.

$$\lambda_C = N \times (N-1) \times \lambda .$$

Ця величина має пуассонівський розподіл. Час обробки запиту розподіляється за експоненціальним законом. І середній час обробки запиту $\mu_C(N)$ обернено пропорційний до масштабу мережі N. З цього випливає, що

$$\mu_C = \frac{K}{g(N)}.$$

Тоді середній час відгуку контролера при централізованій структурі буде

$$E\{T_C(N)\} = \frac{1}{\mu_C - \lambda_C}.$$

В децентралізованій структурі декілька контролерів. Припустимо, що кількість контролерів буде m_D. В першому типі децентралізованної структури кожен контролер має свою локальну мережу. Тому, існують два випадки:

коли весь шлях управляється одним контролером, і коли необхідно задіяти декілька контролерів. Глобальний запит поділяється на два запити: на локальний і на ще один локальний або глобальний до іншого контролера. Якщо середня дистанція контролерів d_D, кожен глобальний запит буде поділений на d_D + 1 запитів. І час відгуку на глобальний запит буде дорівнювати сумі часу відгуку на локальні запити. Відповідно до цього, $\frac{N^2}{m_D} - N$ – кількість локальних шляхів, $N^2 - \frac{N^2}{m_D}$ – кількість глобальних шляхів. Тому швидкість прибуття запитів на кожен контролер буде:

$$\lambda_{D,l} = \lambda \times \frac{(N^2 - \frac{N^2}{m_D}) \times (d_D + 1) + (\frac{N^2}{m_D} - N)}{m_D},$$

а також кожен контролер повинен керувати топологією з $\frac{N}{m_D} + m_D - 1$ вузлами. Тому середній час обробки запиту:

$$\mu_{D,l} = \frac{K}{g(\frac{N}{m_D} + m_D - 1)},$$

а середній час відгуку контролера:

$$E\{T_{D,l}\} = \frac{1 + \frac{N \times (m_D - 1)}{(N-1) \times m_D} \times d_D}{\mu_{D,l} - \lambda_{D,l}}.$$

В другому типі децентралізованної структури, кожен вузол контролеру має повний доступ до мережі. Ця властивість дозволяє розрахувати середній час обробки запиту і середній час відгуку контролера за формулами для централізованої структури. Швидкість прибуття запитів на кожен контролер буде:

$$\lambda_C = \frac{N \times (N-1) \times \lambda}{m_D}.$$

При розрахунках середнього часу відгуку контролера в різних структурах були отримані результати, наведені в таблиці 1.

Таблиця 1

Середній час відгуку контролера в різних структурах

Тип структури	Централізована	Децентралізована (локальна)	Децентралізована (глобальна)
E, c	$1{,}194 \times 10^{-4}$	$1{,}9 \times 10^{-6}$	$1{,}026 \times 10^{-5}$

Для розрахунку бралися наступні дані: кількість вузлів N = 100, швидкість прибуття запитів в секунду $\lambda = 10$, кількість контролерів в

децентралізованих структурах $m_D = 10$ і середня дистанція контролерів $d_D = 5$.

Висновки

Проведений аналіз масштабованості рівня управління SDN, в яких вузли мають подібний обіг трафіку. Проаналізовані стандартні структури рівня управління, а саме – централізована, локально децентралізована і глобально децентралізована з детальним розрахунком, на підставі чого робляться висновки, що локально децентралізована архітектура модуля контролю масштабується з найменшою втратою швидкодії порівняно з іншими.

Література

11. *Nadeau T.* SDN – Software Defined Networks. / T. Nadeau — CA.: published by O'reilly media, 2013. – 80 с.
12. *Goransson P.* Software Defined Networks: A Comprehensive Approach. / P. Goransson. — MA.: Elsevier Inc, 2014. — С. 215.
13. *Azodolmolky S.* Software Defined Networking with OpenFlow / S. Azodolmolky – UK.: Packt Publishing, 2013. – 148 с.

Національний технічний університет України «Київський політехнічний інститут»

НТУУ «КПІ» 1898

Факультет прикладної математики

ПРИКЛАДНА МАТЕМАТИКА ТА КОМП'ЮТИНГ ПМК 2016

Восьма наукова конференція магістрантів та аспірантів

Київ, 20–22 квітня 2016 р.

ЗМІСТ

УДК 004.738

Ст. викладач Дробязко І.П., студент Єрмоленко І.А.
Національний технічний університет України
«Київський політехнічний інститут»

Мобільна пошукова система для динамічного контенту

Abstract

Irina Drobyazko, senior lecturer; Ihor Yermolenko, student

Mobile search system for dynamic content

This paper concerns the task of searching dynamic data. System was developed for quick and convenient adding and searching information, that is periodically changing.

Вступ

В наш час мобільність і цілодобовий доступ до інформації є одними з найважливіших елементів життя. Зростання обчислювальної потужності, відносно невелика вартість, зручний інтерфейс і форм-фактор призвели до величезної популярності мобільних пристроїв та планшетних персональних комп'ютерів. Для сучасної ділової людини мобільний телефон з доступом в мережу Інтернет - головний інструмент в організації та плануванні різних видів робіт.

Оскільки Інтернет є головним способом комунікації у XXI сторіччі, він використовується не тільки для організації та обговорювання робочих справ, а й для спілкування з близькими і друзями, однодумцями, пошуку знайомих за інтересами чи хобі.

На сьогоднішній день існує велика кількість систем для пошуку статичних даних. Але жодна з них не адаптована для пошуку динамічної інформації, тому використовувати їх для роботи з даними, що постійно змінюються, важко або зовсім неможливо.

Постановка задачі

Тому метою роботи є розробка мобільної системи, яка дозволяє працювати з контентом, що динамічно змінюється. Для полегшення

користування даною системою важливо розробити зручний інтерфейс користувача, з використаним сучасним API.

Фактично, не існує додатків чи сервісів, які б полегшували створення і оголошення, і, що більш важливо, пошук динамічної інформації. По-перше, існуючі додатки і сервіси розраховані, як вже було зазначено вище, на обробку статичної інформації. Тобто, ніякої взаємодії чи зворотного зв'язку з людьми, які цікавляться даною інформацією, немає. Безумовно, в деяких випадках важливіше враховувати витрати, а поспілкуватися с кожною людиною і вислухати побажання кожного просто неможливо. По-друге, більшість схожих за призначенням сервісів є платними і коштують достатньо дорого. Отже, можна зробити висновок, що, на перший погляд схожі системи розроблені для зовсім різних цілей. Частково вирішує дану проблему можливість ділитися інформацією в деяких соціальних мережах, проте це не дуже зручно.

Таким чином, розробка мобільної системи, яка буде використовуватися для пошуку динамічних даних, є надзвичайно актуальною задачею.

Термінологія

Статичний контент - це незмінна інформація, яка, як правило, мфє постійну адресу в Інтернеті. На сайті це може бути, наприклад, розділ «Про компанію», загальне поняття, авторська стаття.

Динамічний контент - на відміну від статичного, представляє собою інформацію, яка часто змінюється.

Опис функцій

Мобільна пошукова система розроблена у вигляді додатку для мобільних пристроїв під управлінням операційної системи Android. Прикладна програма або додаток – програма, призначена для виконання певних завдань і розрахована на безпосередню взаємодію з користувачем. З чинників, які вплинули на вибір платформи, можна відмітити інтеграцію з сервісами Google, популярність даної ОС. Окрім того Android, на відміну від основних своїх конкурентів (iOS, Windows Phone), є відкритою платформою, що дозволяє реалізувати на ній більше функцій і значно спрощує розробку системи.

Створений додаток представляє собою мобільний event-менеджер, тобто програму для організації заходів. На відміну від існуючих програм, що

схожі за призначенням, додаток орієнтований в першу чергу на додавання і пошук інформації про неофіційні заходи, яка, як правило, є динамічною (швидко змінюється та легко додається). Для полегшення користування реалізована можливість роботи з картами місцевості за рахунок інтеграції стандартних сервісів Google.

З огляду на особливості операційної системи Android, обмін інформації між сервером і додатком відбувається за рахунок обміну між ними JSON-файлами [1]. Така схема є стандартною для багатьох web-сервісів, оскільки файли цього формату є незначними за розміром, добре придатні для серіалізації складних структур і прямо інтерпретується за допомогою JavaScript в об'єкти.

Однією з основних можливостей програми є додавання інформації про власний захід. Для цього користувачу необхідно пройти реєстрацію. Таким чином, потенційні учасники заходу можуть бачити, хто організовує захід. Не дивлячись на те, що програма розроблена як планувальник неофіційних заходів, це не означає, що всі учасники будуть знати організатора (наприклад, якщо це збори клубу за інтересами). Тому інформація про організатора вкрай необхідна.

При пошуку схожих за своїми функціями сервісів, можна зрозуміти, що жоден з них не має можливості відмічати захід на географічній карті. Тому, головною перевагою розробленого сервісу є можливість працювати з картами Google Maps. Користувач має змогу залишити опис заходу і додати відмітку на карті місцевості. Для реалізації даної функції застосовуються стандартні засоби Android та Google Maps API, завдяки якому програма може використовувати Google Maps карти [2]. API автоматично обробляє доступ до Google Maps серверів і завантажує дані. Також API можна викликати для того, щоб додати маркери, багатокутники і накладки, різні шари та типи карт, для зміни виду відображення карти користувача в конкретній області карти. Ключовим об’єктом класу в даній API є MapView, який відображає карту з даними, отриманими від служби Google Maps. Додаток може використовувати методи класу MapView і контролювати карту програмно, приєднати tool bar з кнопками та елементами відображення в верхній частині карти. Google Maps API не включені в Android платформу, але доступні на будь-якому пристрої з Google Play, з версії Android 2.2 або вище, використовуючи Google Play services.

При занесенні заходу необхідно вказати дати початку і закінчення, тип запланованого заходу, його назву. Також необхідно зазначити чи буде інформація про захід доступна всім (відкритий захід), чи доступ до нього отримають тільки запрошені особи (закритий захід).

Окрім того, додаток визначає місцеположення користувача. Дана можливість необхідна як при додаванні інформації про подію на карту, так і

при пошуку місця проведення заходу. Наприклад, при додаванні заходу можлива ситуація в якій незручно вводити адресу вручну, тому слід використовувати автоматичне встановлення місця. При пошуку необхідного заходу, можна продивитися на карті дорого до нього, та обрати маршрут за яким легше пройти.

Використовуючи геолокаційні сервіси із Google Play, додаток може отримати останнє відоме місце розташування пристрою користувача. У більшості випадків, потрібно знати поточне місцезнаходження користувача, яке, як правило, еквівалентне останньому відомому розташуванні пристрою. Зокрема, використовується об'єднання різних геолокаційних провайдерів для визначення місцезнаходження користувача. Об'єднаний геолокаційний сервіс є одним із location APIs в Google Play services. Після того, як додаток підключений до Google Play services і locations services API [3], він може отримати останнє відоме місце розташування пристрою користувача. Коли додаток підключений до цих служб, він може використовувати доступний провайдер місцезнаходження який видає розташування пристрою. Точність розташування, яка надається цим провайдером, визначається параметром, який записаний в маніфесті програми. Провайдером місцезнаходження може бути або GPS, або мережа. Для отримання даних про місцезнаходження через мережу, необхідно мати доступ до Інтернету через мобільного оператора чи Wi-Fi. Отримання даних через мережу, як правило, більш точне і використовує менше заряду акумулятора, проте для цього необхідно завжди мати підключення. Тому, програма має змогу використовувати обидва провайдера

Іншою основною функцією програми є пошук заходів, яка також є унікальною для даної системи. В схожих додатках, людина може дізнатися про захід тільки після запрошення організатора. Окрім того, відсутня можливість подивитися на карті місцевості інформацію про місце проведення заходу.

Пошук реалізовано за допомогою вже вказаних вище технології. Варто відмітити, що програмою можна користуватися без реєстрації. Проте для додавання власного заходу, чи отримання запрошення на закриті користувач повинен бути зареєстрованим в даній системі.

Для пошуку заходів використовуються спеціальні фільтри, щоб користувач мав змогу обрати дату чи тип заходів, які його цікавлять.

Після введення фільтрів, на карті можна побачити всі заплановані зустрічі і подивитися інформацію про них.

Висновки

Проаналізовані загальні вимоги до мобільної пошукової системи, що вирішує задачу обробки динамічного контенту.

Розроблений один з варіантів реалізації системи з функціями додавання і пошуку, використанням карт і місцем знаходження користувача, у вигляді мобільного додатку зі зручним інтерфейсом. Розглянуті необхідні засоби для побудови даного сервісу, способи роботи з картами, способи визначення місцезнаходження.

Література

1. *Ленгсторф Д.* JSON: що це, як це працює і як використовувати це / Д. Ленгсторф // copterlabs.com: інтернет-журн. - [Електроні дані] - 2009. - Режим доступу: http://www.copterlabs.com/json-what-it-is-how-it-works-how-to-use-it/ - [Текст з екрану] - Дата звернення: 13.03.2016.
2. *Майер. Р.* Android 4. Программирование приложений для планшетных компьютеров и смартфонов / Р. Майер - М.: Эксмо, 2013. - С. 535-578.
3. *Коматинени С.* Android 4 для профессионалов. Создание приложений для планшетных компьютеров и смартфонов / С. Коматинени, Д. Маклин - М.: Вильямс, 2012. - С. 435.